The MagPi

Issue 106 | June 2021 | magpi.cc

The official Raspberry Pi magazine

RASPBERRY PI BIG BUILDS

Gigantic gadgets to make in your garage

PLUS! Summer Projects

Top 10 media players

Build a burglar alarm

Discover an amazing drinks dispenser

50 PAGES OF PROJECTS & TUTORIALS

WELCOME
to *The MagPi* 106

This month's edition of *The MagPi* magazine is all about massive projects. Rob has discovered the big builds that take time, space, and attention. The result is always special. So super-size your projects with our Big Builds feature (page 36).

We had a lot of fun this month and we hope you enjoy our big builds. And if you want something a bit more manageable, then we have got you covered. I love Picoth (page 14), it's a Raspberry Pi Pico keyboard project that acts as a two-factor authentication device. There's also an amazing Drinks Machine (page 26), Tiny Mac (page 28), and ShouldI: an eInk display that taps into the energy network and lets you know when it's environmentally friendly to power up your gadgets (page 22).

Nik has written an amazing Summer Projects feature (page 72). Like a lot of people, we've rediscovered a love of hiking, camping, and the great outdoors. Nik's feature is full of ideas for using Raspberry Pi in the wild.

Aside from a few spots of rain, we've had a lot of fun looking at big builds and outdoor projects. I hope the sun shines, and you have a lot of fun with Raspberry Pi this month.

Lucy Hattersley Editor

EDITOR

Lucy Hattersley

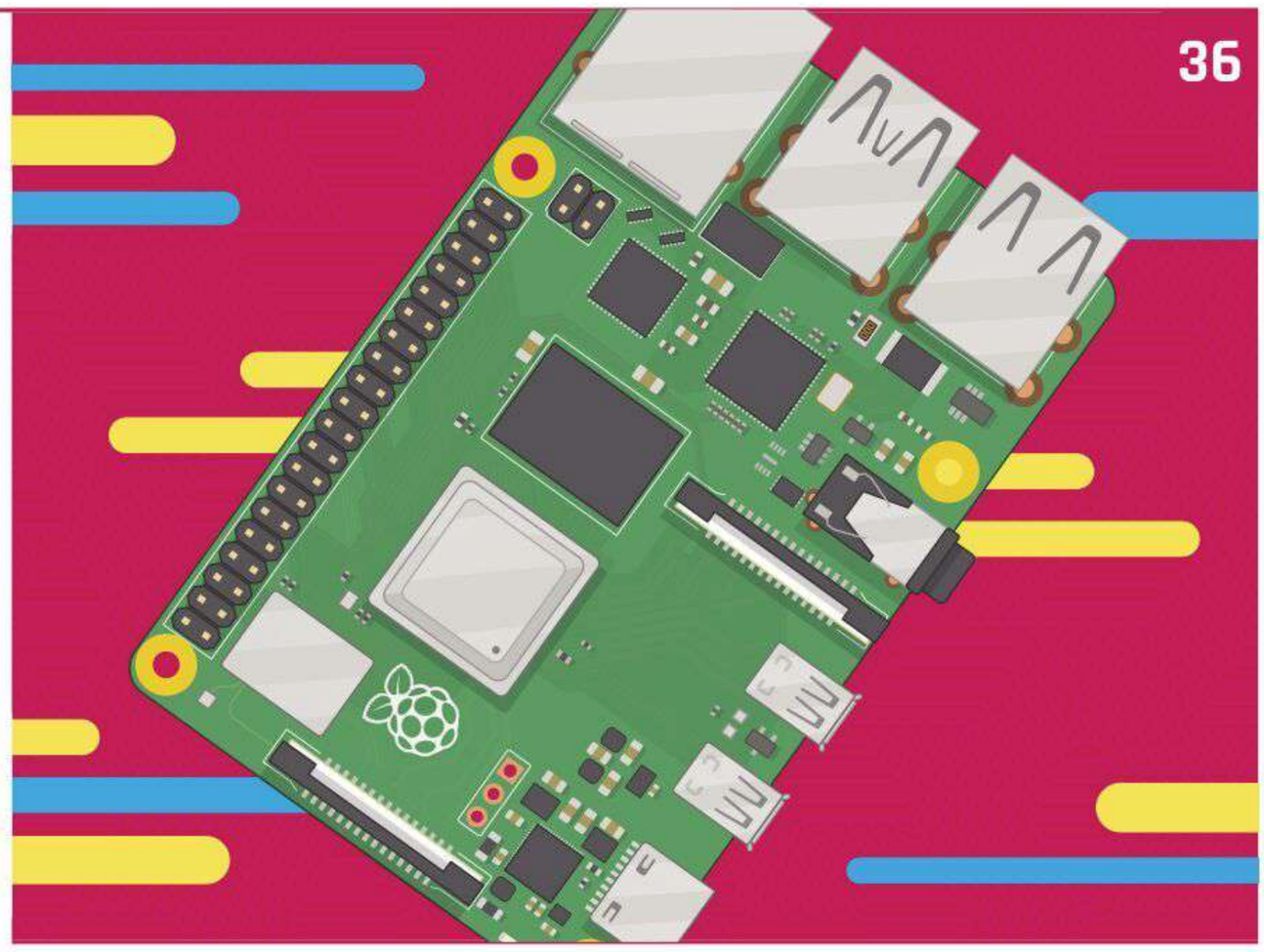
Lucy is editor of *The MagPi* and has foolishly signed up for the Brighton Marathon.

@LucyHattersley

Contents

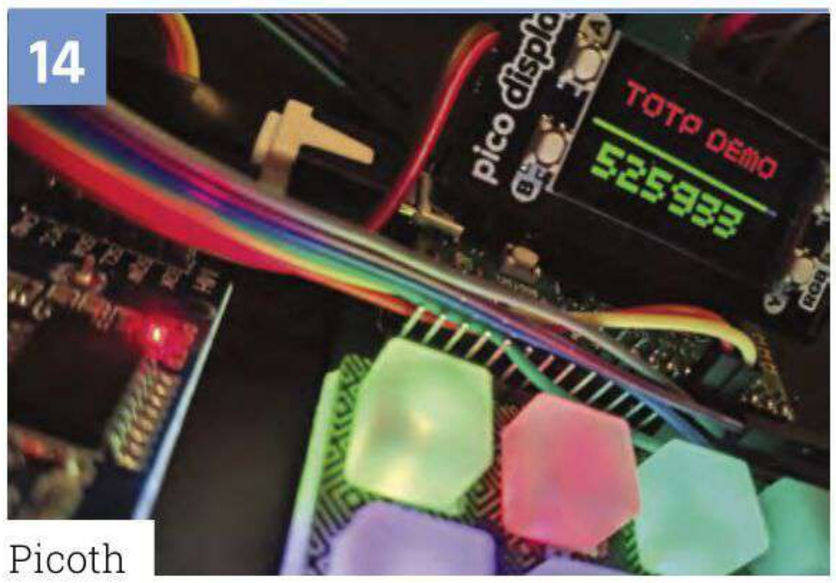
Issue 106 June 2021

Cover Feature

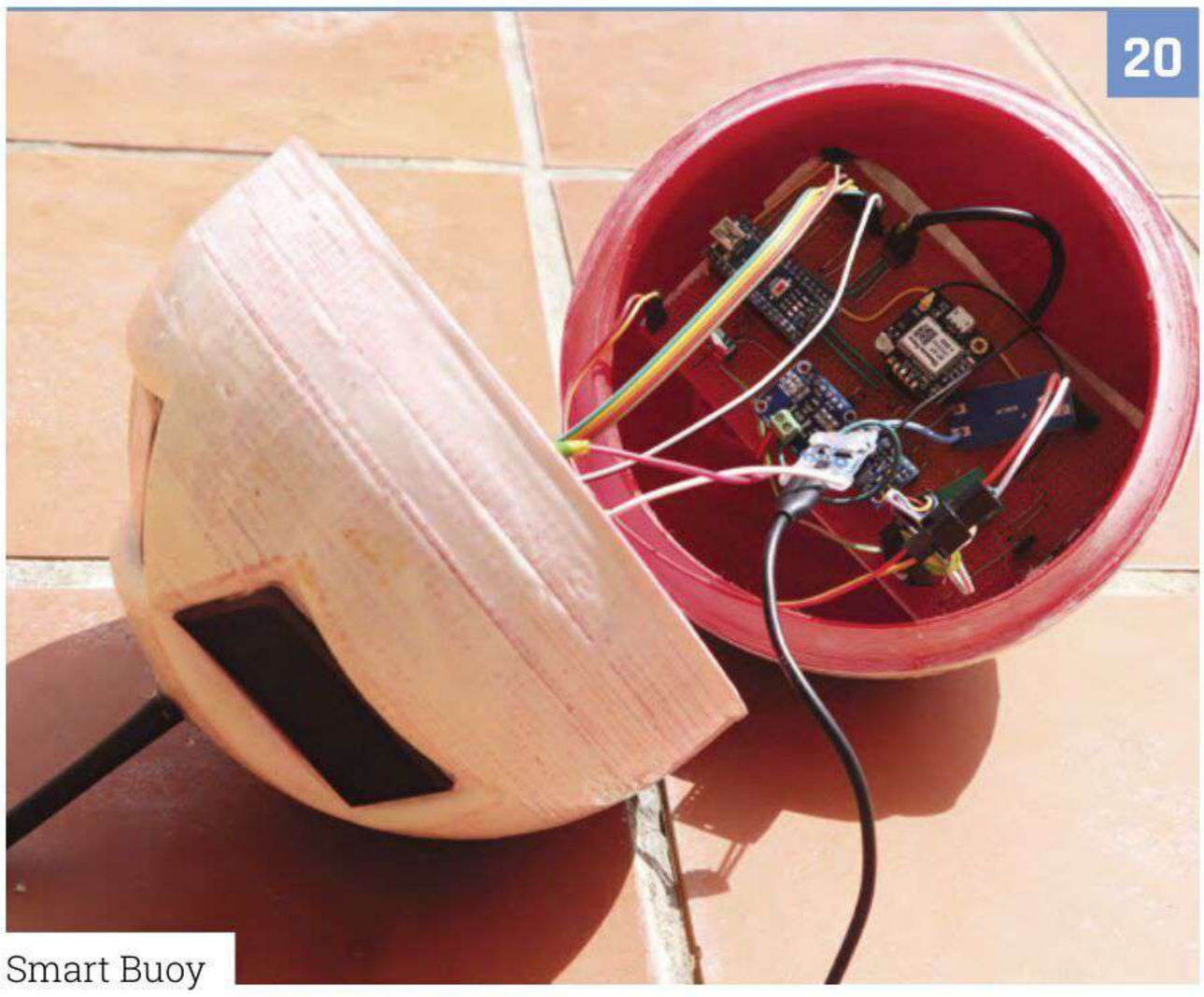


Regulars

Project Showcases



Picoth



Smart Buoy

Tutorials

The Big Feature

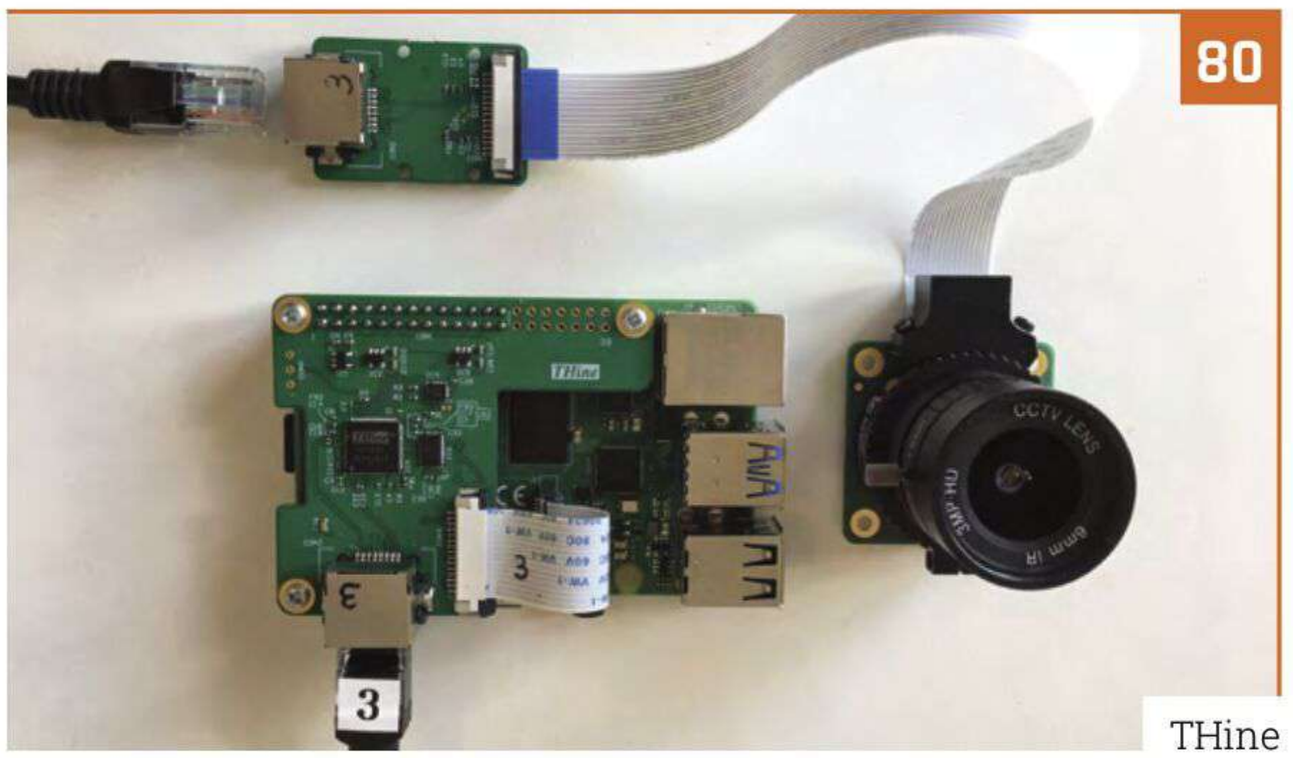
Summer Projects

Reviews

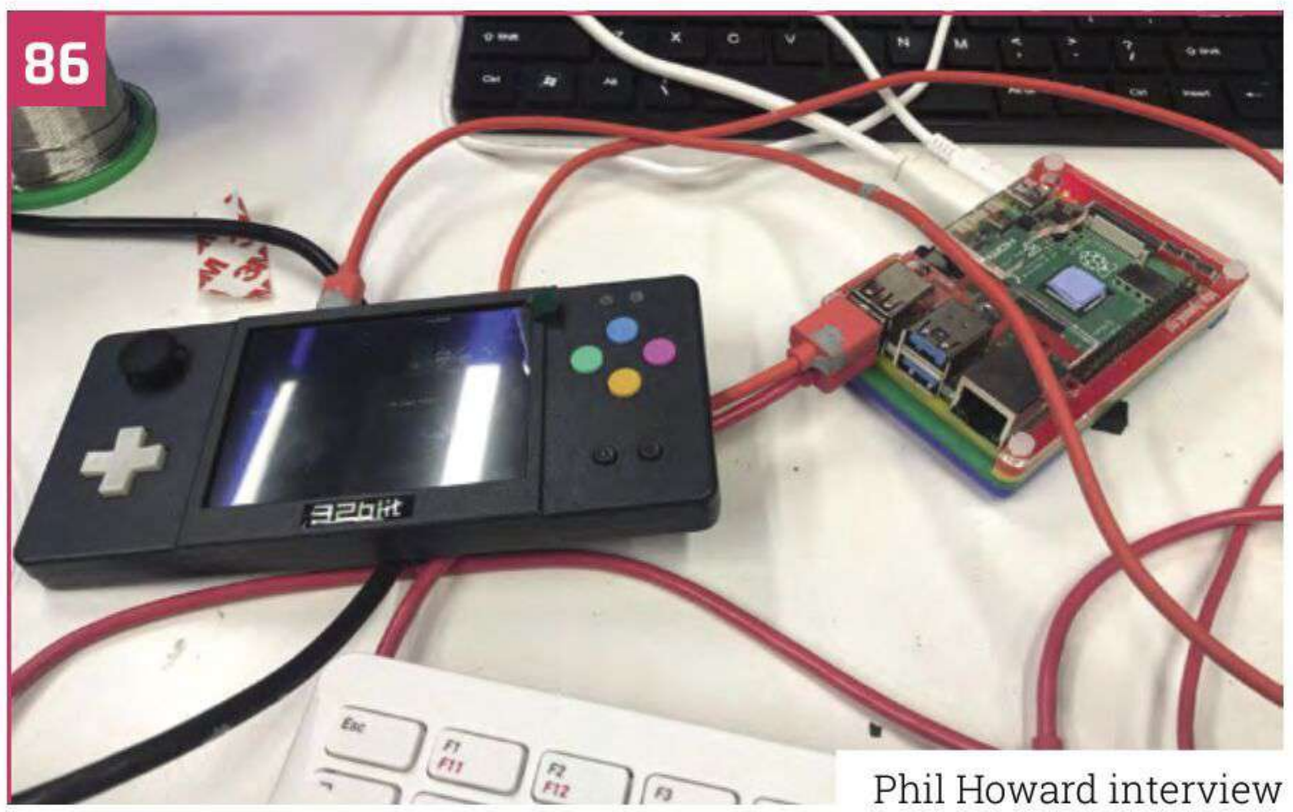
Community

Build an arcade machine - part 3

Pico-Voice - part 1



THine



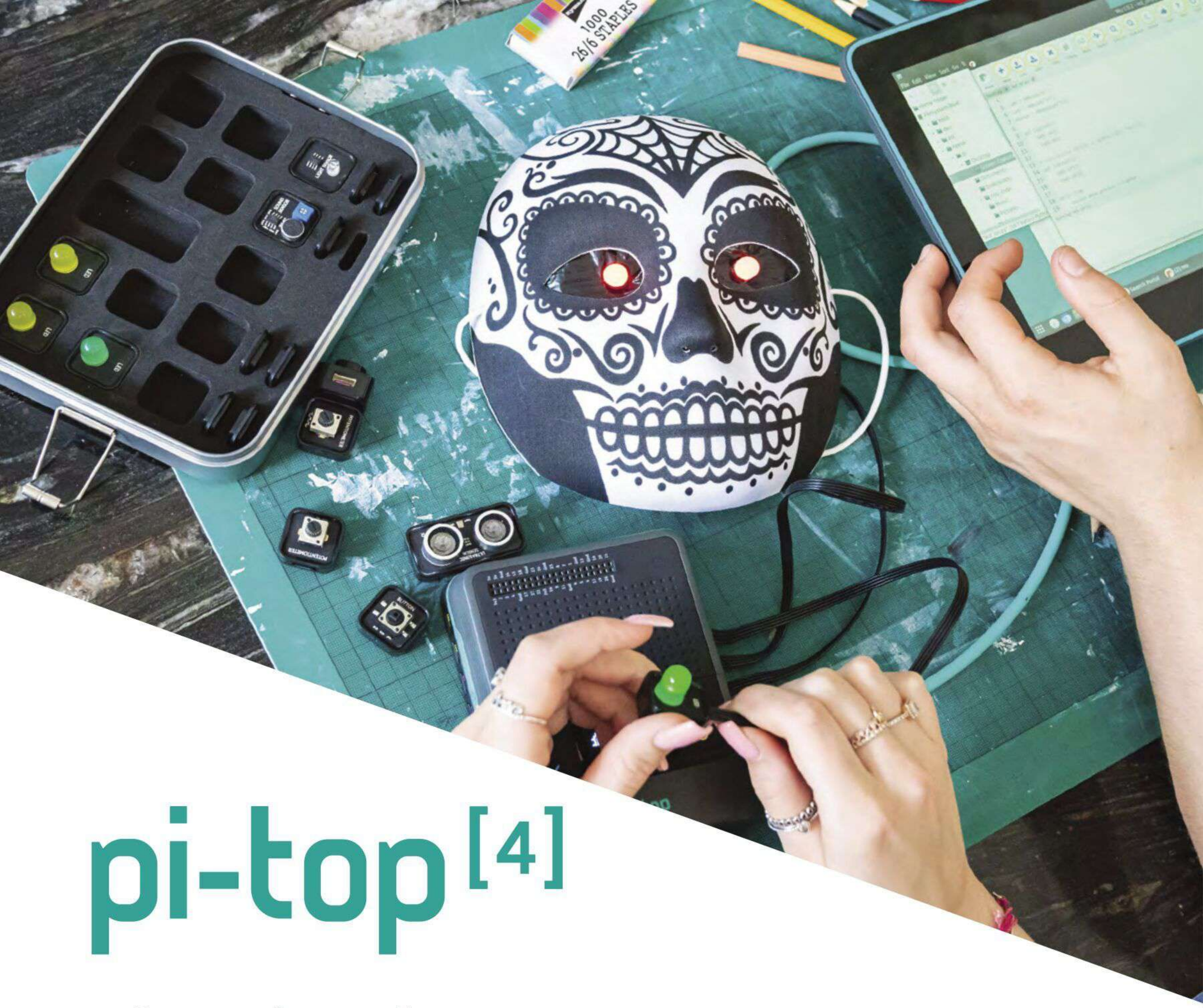
Phil Howard interview

WIN 1 OF 5 GROVE STARTER KITS FOR **RASPBERRY PI PICO** 95

Convert Raspberry Pi 4 to a tablet with RasPad 3.0

- Fully compatible with Raspberry Pi 4
- 10.1” IPS touchscreen with an impressive resolution of 1280x800 pixels
- Full access to all Raspberry Pi GPIO, Ethernet, HDMI, audio, USB, and power ports
- Compatible with Raspberry Pi OS, Arduino, Ubuntu Desktop, Chromium OS, Android and more
- 10-point multi-touch screen, 5-hour battery life
- Easy to assemble

Buy now! raspad.com/products/raspadv3



pi-top [4]

The simplest way to get started with the Raspberry Pi

Everything you need to power up your projects is pre-installed in this rugged case, including the Raspberry Pi 4 itself.

With the modular pi-top [4] computer you can create anything from a musical instrument to the ultimate alarm system. Explore our online project library to learn to code and control your creations.

pi-top We make the future.

Gain skills for the future

One-to-one courses to learn coding in Python and digital making:

ONLINE COURSE

Coding Minecraft with pi-top [4]

Learn how to program the game you love to play every day.

ONLINE COURSE

Coding Music with pi-top [4]

Learn to program your very own music Synth using Sonic Pi.

Explore the exciting world of physical computing and coding with our one-to-one courses. You'll be guided by pi-top Certified Trainers, chosen from top universities.

- Perfect for ages 11-17
- Beginner to intermediate level
- 12 one-to-one lessons booked at your convenience
- Siblings can join for free!

FREE

pi-top [4] computer and Sensor Kit worth over £250 included with every course!

Find out more at **pi-top.com/MagPi**

Farm Sensor Dashboard

How is IoT helping farmers in Essex, New York? Its creator talks to **Rob Zwetsloot** about it

MAKER

Léo Galley

An independent UX designer and full-stack engineer from Switzerland, now based in Brooklyn, NYC.

magpi.cc/farmsensor

As we write this, the sun is shining and is reminding us of going strawberry picking in summers past. It's a fun thing to do, especially when you have some cream to enjoy them with later. Léo Galley, from Brooklyn, found himself picking crops himself.

"After helping a friend doing farmer's produce deliveries around Brooklyn, I reached out to Essex Farm, and I was kindly invited to come up there and see for myself how they work," Léo explains to us. "While picking tomatoes, radishes, cucumbers, and more in the unforgiving August sun, I noticed someone with a temperature gun going between all the greenhouses and the fridges, to monitor the temperature. And so I had an idea – and a few months later, a prototype."

The result ended up as the Farm Sensor Dashboard (**farmsensordashboard.com**), which not only helps the farmers track the temperature of these greenhouses and fridges, but also the humidity.

Simple build

Anyone who has built soil sensors or worked with temperature sensors knows that this is quite a simple build, and in fact only really requires one main sensor, a DHT22. To get it fully connected and

Storage areas require specific conditions to make sure produce stays fresh

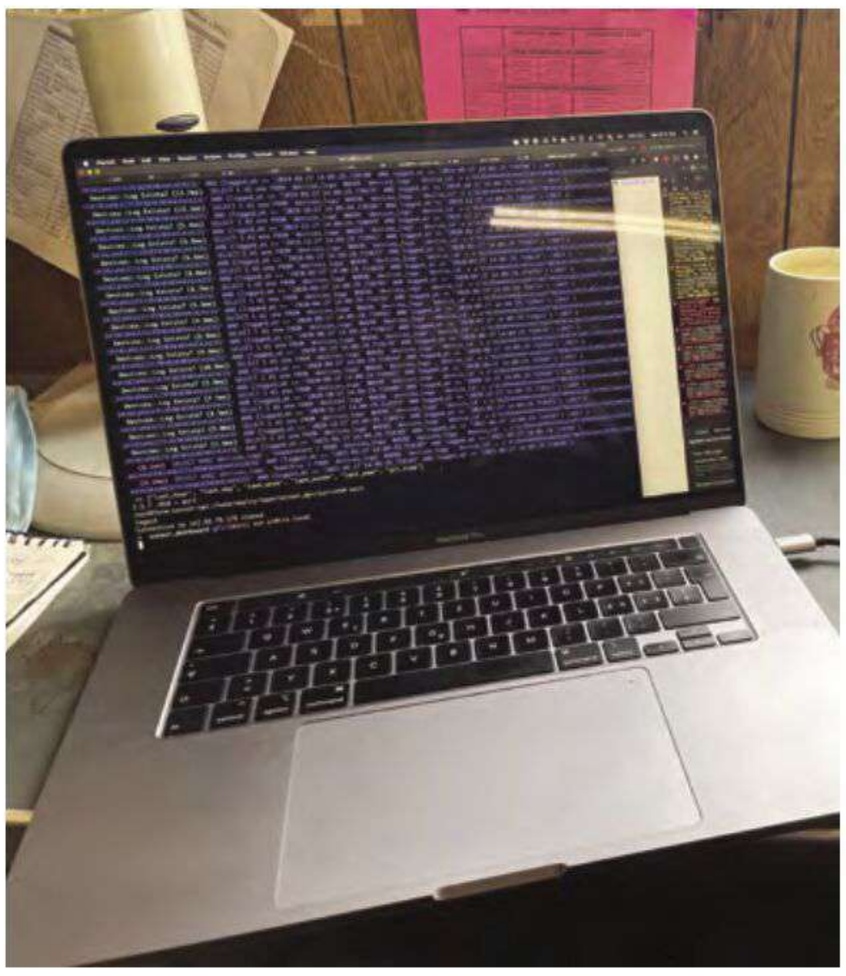
One of the many Raspberry Pi Zero boards that measure conditions in the areas around the farm

Quick **FACTS**

- Each sensor has its own name
- If you don't remember which Raspberry Pi goes where, the space is named as well
- The DHT22 sensor can measure between -40 and 80°C (-40 to 176F)...
- ...Perfect for the freezers, fridges, and greenhouses around the farm
- There are 35,000 farms in the state of New York, which take up 20% of the land

▶ Power to the devices is an important consideration

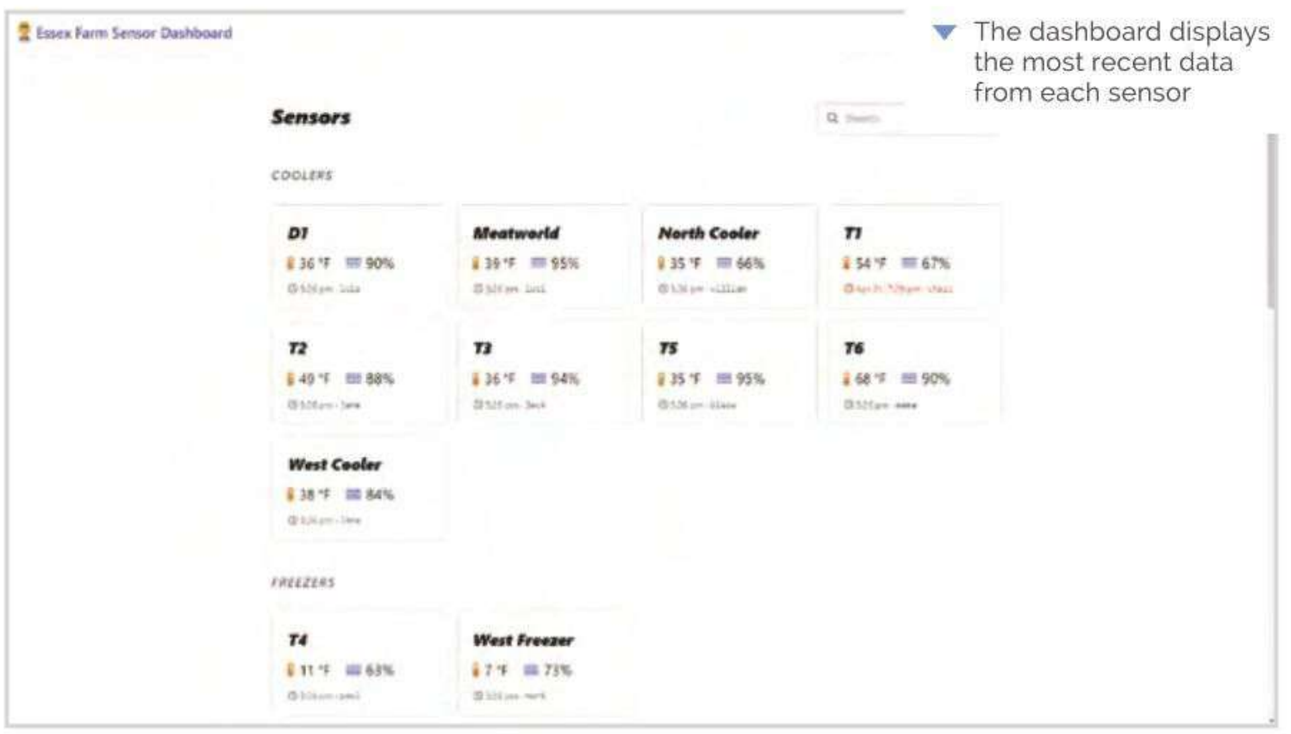
▼ Setup is simple for each device



> The farmers find the app useful: they went from checking spaces twice a day to once a day

able to work with the online platform, Léo turned straight to Raspberry Pi. "Raspberry Pi Zero W met all my requirements: wireless LAN, GPIO for the sensor, cheap, low specs, and of course the immense availability of online resources and supportive community," he says.

The dashboard itself is very easy to use, giving you the latest readings at a glance and allowing you to drill down into historical measurement graphs. Data like this is incredibly useful for efficiently using humidifiers, watering, and refrigeration systems – it could even be used to automate those



▼ The dashboard displays the most recent data from each sensor

▼ The basic build of the devices used by Farm Sensor Dashboard

▲ The devices hang in their respective spaces within a sealed bag

kinds of mechanisms. It runs on a simple Python script, while the dashboard website uses a mixture of Rails, Next.js, React, and some custom CSS work from Léo himself.

"The farmers find the app useful: they went from checking spaces twice a day to once a day," Léo reveals.

More automation

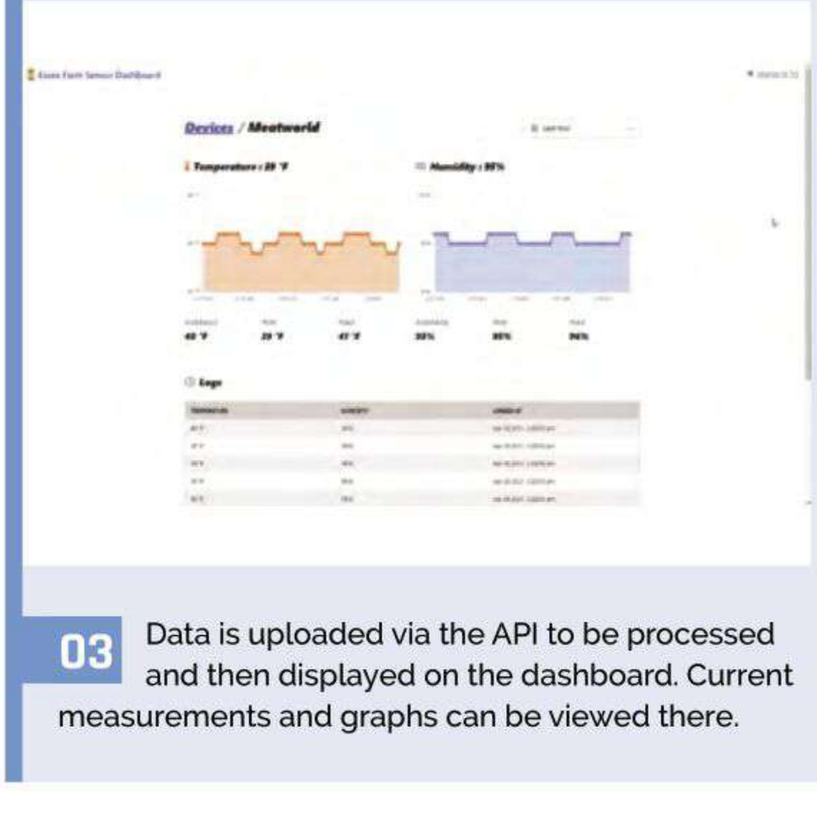
As well as the current temperature and humidity sensing, Léo and the Essex Farm folks plan to upgrade the sensor suite on the current dashboard: "We're going to add more sensors, and I'm looking at other problems such as tracking soil moisture, tracking organic matter, and possibly use a solar-powered WiFi repeater."

It's an interesting system, and we hope Léo and the rest of the Essex Farm team manage to streamline their growing. If you want to check out the farm itself, you can head to **essexfarmcsa.com**, and see what kind of produce is being created with the help of Raspberry Pi.

Tracking crops

01 The setup with Raspberry Pi Zero is put in a location where the sensors can be useful, and also in range of the wireless network.

02 Every minute, a simple Python script is run that reads the temperature and humidity from the sensors.



03 Data is uploaded via the API to be processed and then displayed on the dashboard. Current measurements and graphs can be viewed there.

Picoth two-factor authentication device

Protecting your digital assets and identity are the goals of one maker's Raspberry Pi Pico-based security project. **Rosie Hattersley** gets a safety briefing

MAKER

Angainor

Angainor was attracted by all manner of science and technology from an early age. His interests include cryptocurrency, 3D printers, maths, and powerful microcontrollers.

@Angainor15

This compact project has few components and costs a modest 40 euros

With a self-described "young and curious mind", maker Angainor is a veteran of Raspberry Pi projects ranging from routers and home automation to retro gaming and time-lapse photography. There are "so many projects out there, you just want to buy 50 Raspberry Pi [boards] and build them all," he enthuses. Rattling off his list of previous builds, he declares, "I'm bound to have forgotten some of them." When Raspberry Pi announced Pico, its new microcontroller, he knew the time had come to create Picoth, a tiny security device he'd use all the time.

Simple security

Picoth is a "small USB keypad with RGB buttons and a nice colour TFT screen. Just plug it in and you get a powerful authentication assistant that will type in your 2FA (two-factor authentication) codes for you. You can store up to ten codes per page, with any number of pages you need," making it ideal for online banking, GitHub, Twitter, and messaging platforms. Rather than having to unlock the phone, open the authenticator app, scroll to find the code, then type it in within a few seconds, Angainor says Picoth is set up with one touch to display the code with its label and one touch to auto-type it. Furthermore, the screen displays the remaining time, since 2FA codes change every 30 seconds.

"This first goal of the project was to have something I feel the need for every single day: a small and trustable device that can keep my various 2FA authentications safe and always at hand," says Angainor of why he created Picoth. Raspberry Pi Pico "handles the hardware – a 4×4 matrix keypad and its 16 RGB LEDs, the 240×135 TFT colour screen, and a clock module – as well as all the software: code generation, USB_HID emulation, and animations".

With experience using Python on other microcontrollers, Angainor jumped at the chance

The two-factor authentication device displays the randomly generated login code along with a yellow bar indicating the time remaining to use it

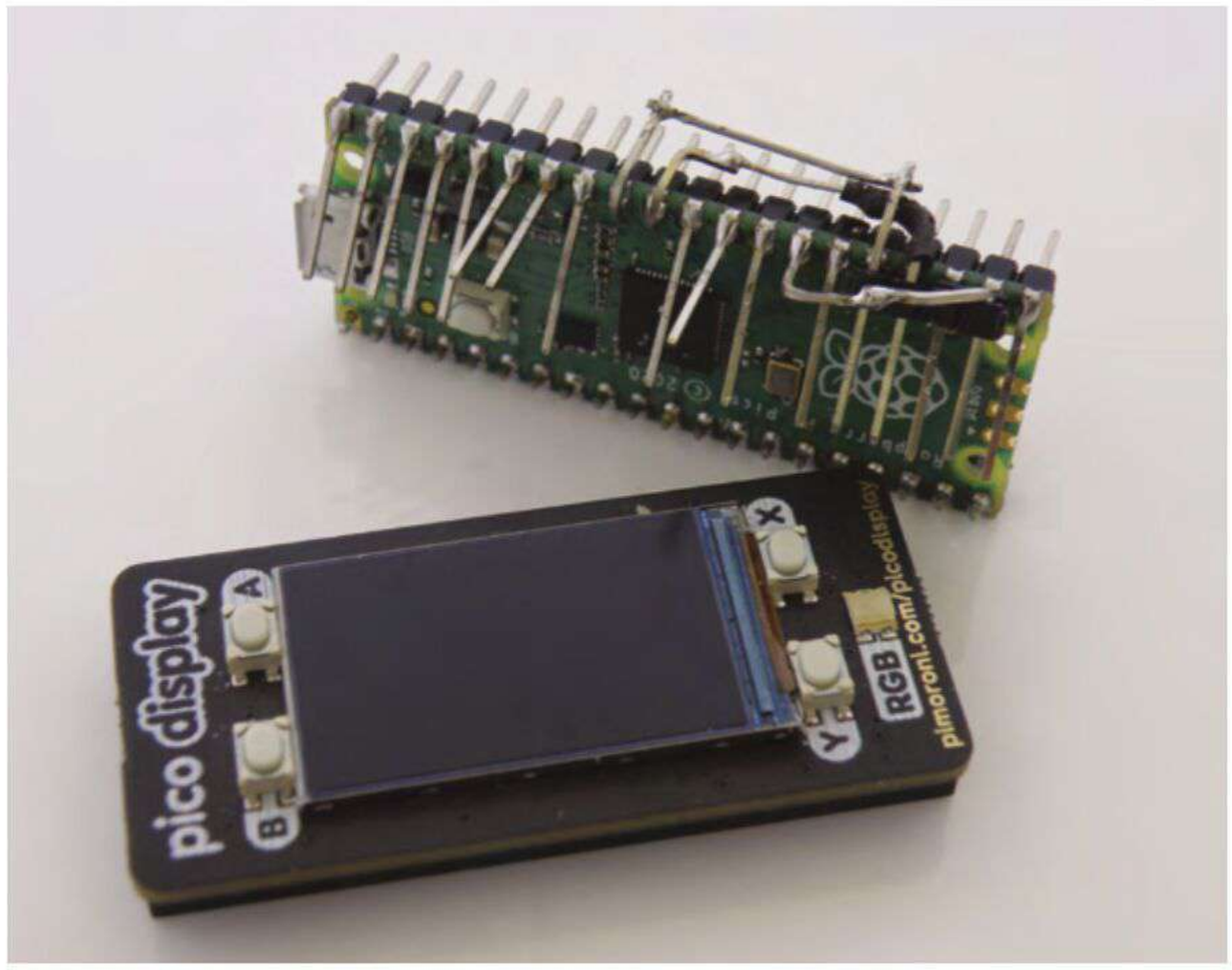
Raspberry Pi Pico proved an ideal basis as extensions and packs were readily available

A Pimoroni Pico RGB Keypad Base made for a visually interesting security input device

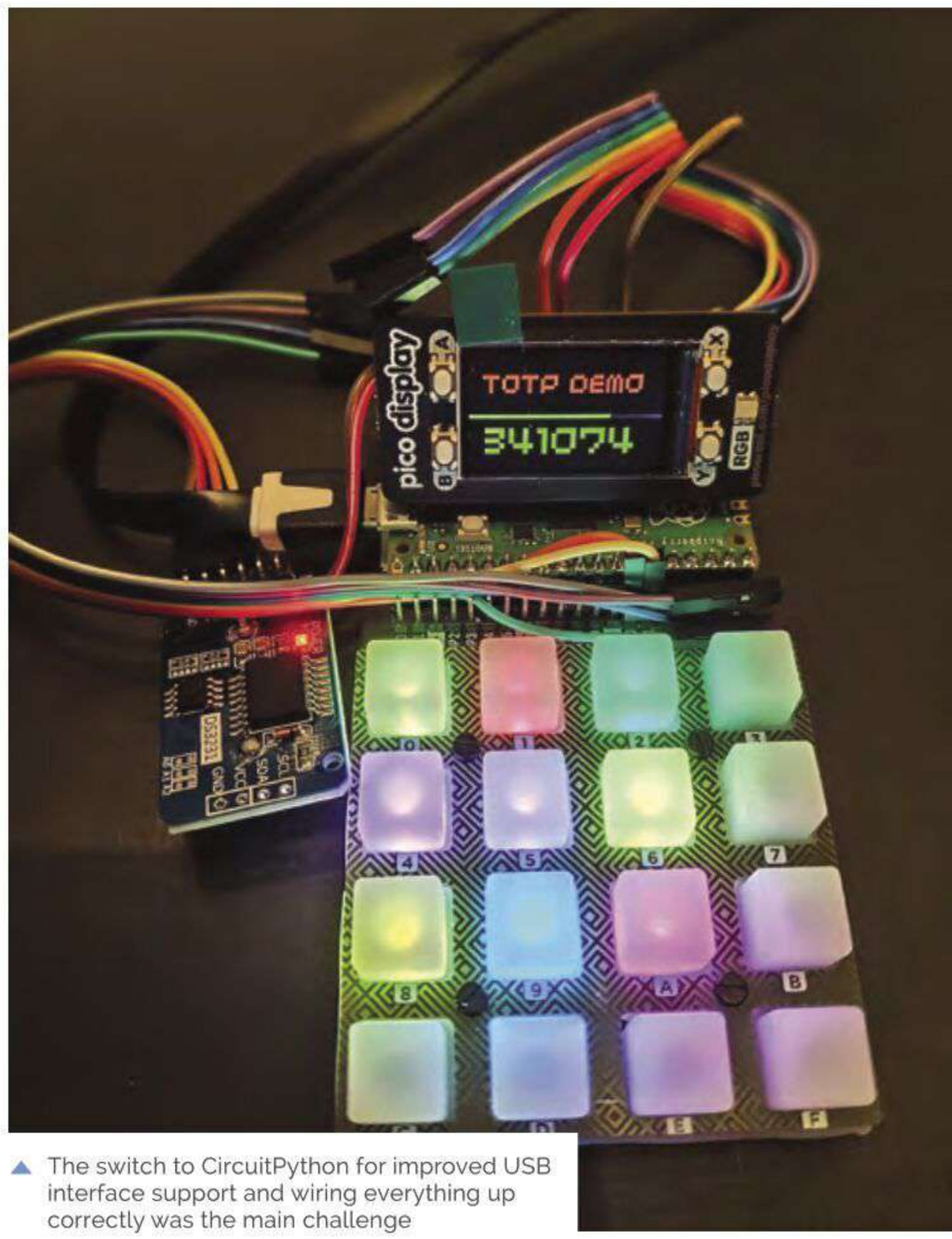
Quick **FACTS**

- Angainor says Pico packs and Python libs are "invaluable"
- His "potentially time-consuming DIY project" took two days
- The project cost €40 to build
- He urges wannabe makers to "just do it!"
- "...You'll be amazed by the community help and feedback"

> "I just love Python's expressiveness and compactness, and it's so seamless to prototype with"



▲ A certain amount of adjustment to the header pins was needed

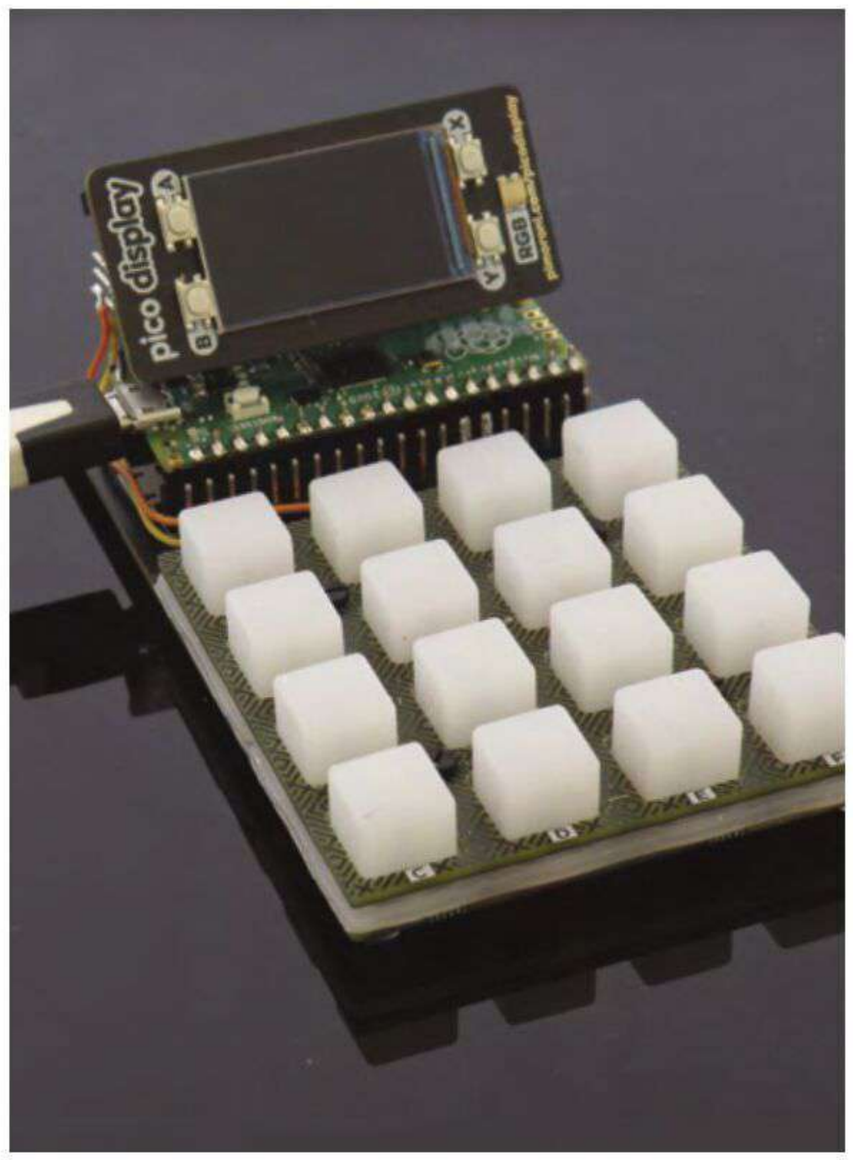


▲ The switch to CircuitPython for improved USB interface support and wiring everything up correctly was the main challenge

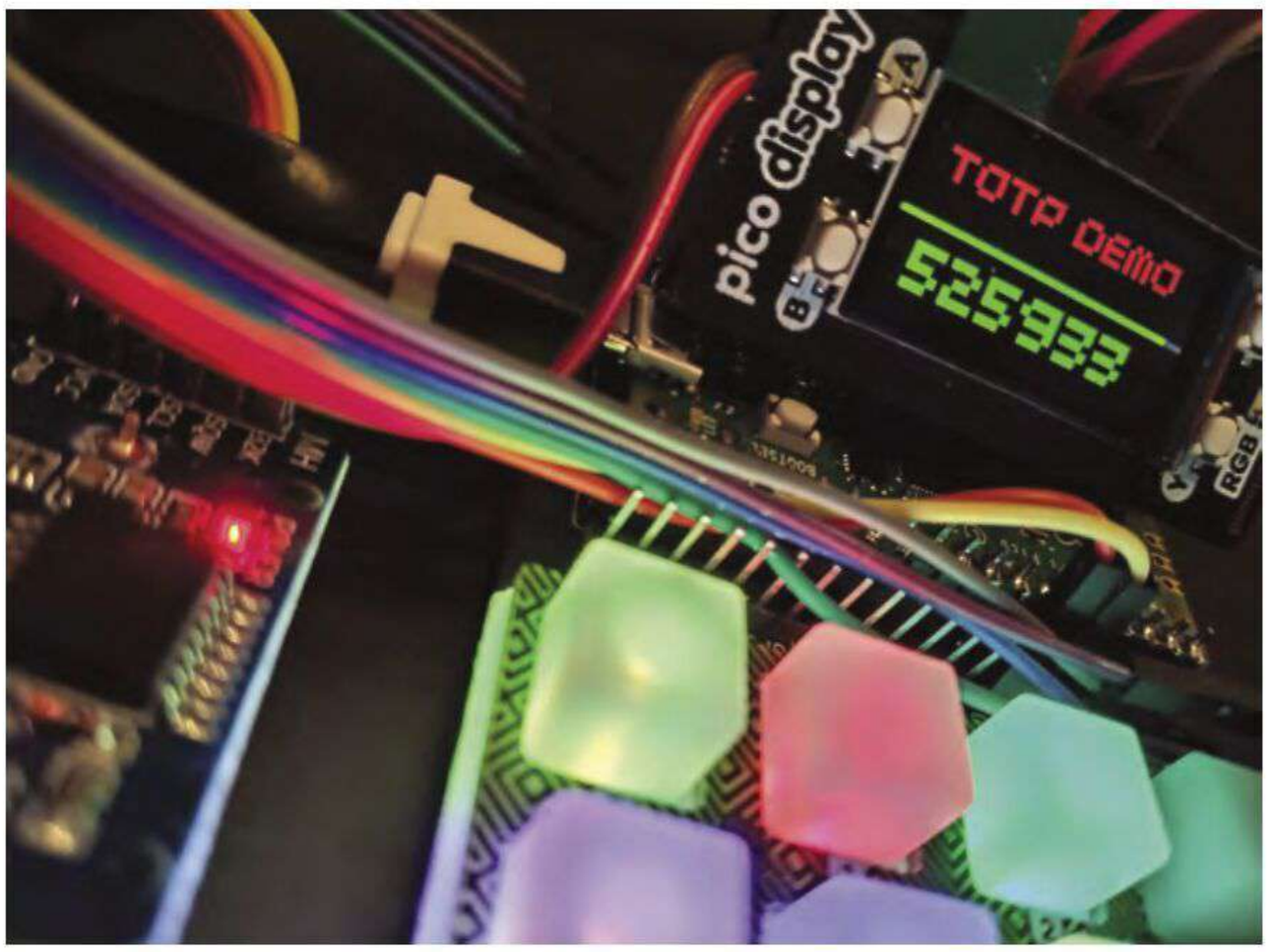
to do the same with a Raspberry Pi one. "I just love Python's expressiveness and compactness, and it's so seamless to prototype with." He likes Pico because it boots in a couple of seconds, is very affordable, and hardware vendors had extensions and packs available as soon as Pico went on sale. He chose Pimoroni's "intelligent" multifunction Keypad Base, having tried to create a similar device using the firm's Keybow.

Tricky code words

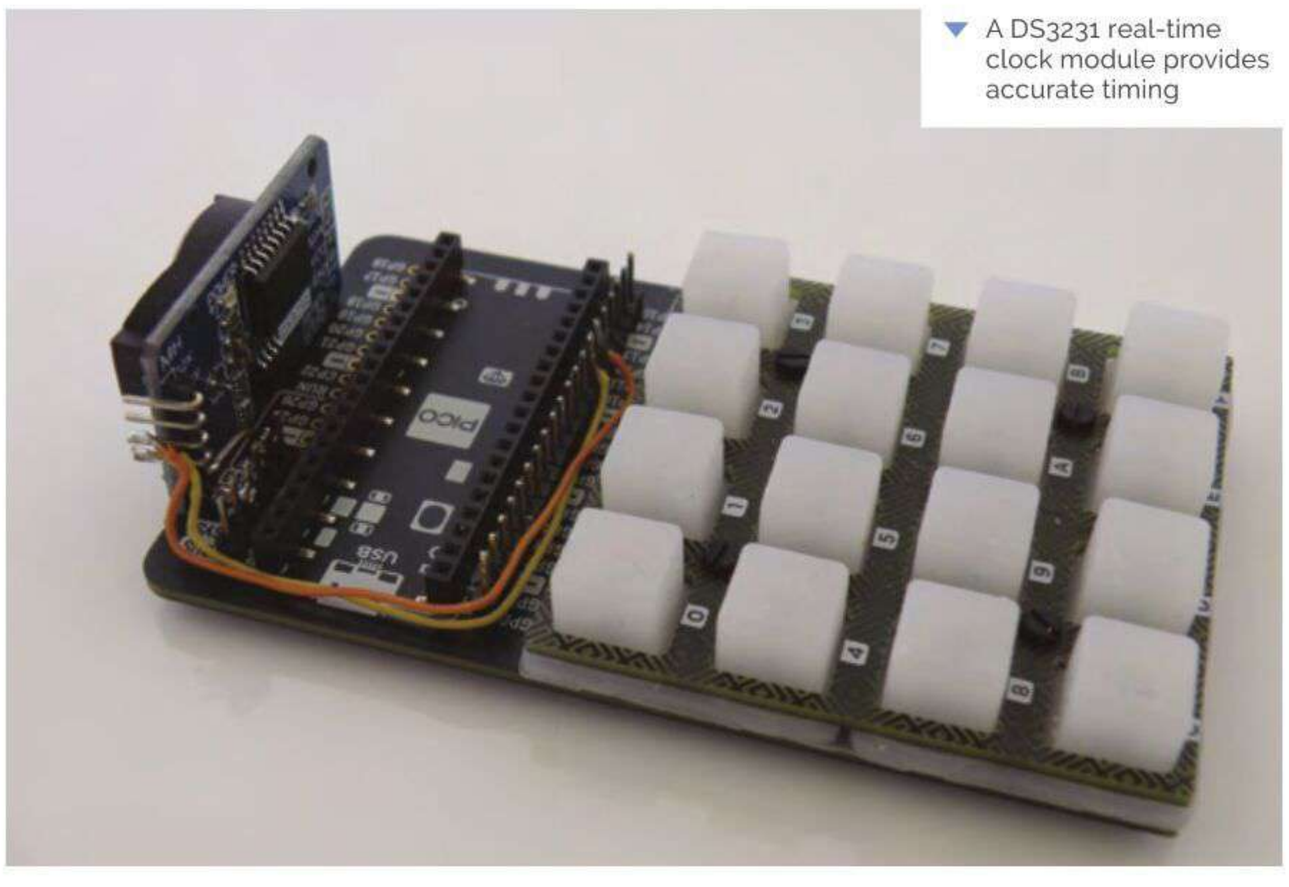
For Picoth, he says the "fun thing was parsing the official documents to check the flexible GPIO features, and what pins could be routed to I2C and SPI ports." Having connected Pico to the display and keypad, he set about coding using MicroPython and Pimoroni libraries. He compiled everything himself as the Pimoroni firmware lacked the SHA-256 and SHA-1 he needed, editing the display library code since the pins were hard-coded. He got his device to work but hit a snag (see **magpi.cc/harshdecision**) relating to the USB input devices. It meant the device couldn't type the code itself. "While usable, this was a serious drawback" and prompted a move to CircuitPython which is slower but has built-in USB-HID support.



The change meant "significant changes and added constraints" to his plans. Getting the keypad and display to work together was perhaps the biggest hurdle in the project but, as ever, the Raspberry Pi community came up trumps: "an awesome library by Sandy Macdonald (**magpi.cc/keybow2040**), targeting a keybow2040, that was a perfect fit for the keypad," says Angainor. He also notes that Raspberry Pi and Python's ecosystems were what made the project possible at all, turning things into a matter of deciding how to assemble the parts. "There were some hacks needed because some items were not to be used that way, but that's part of the fun."

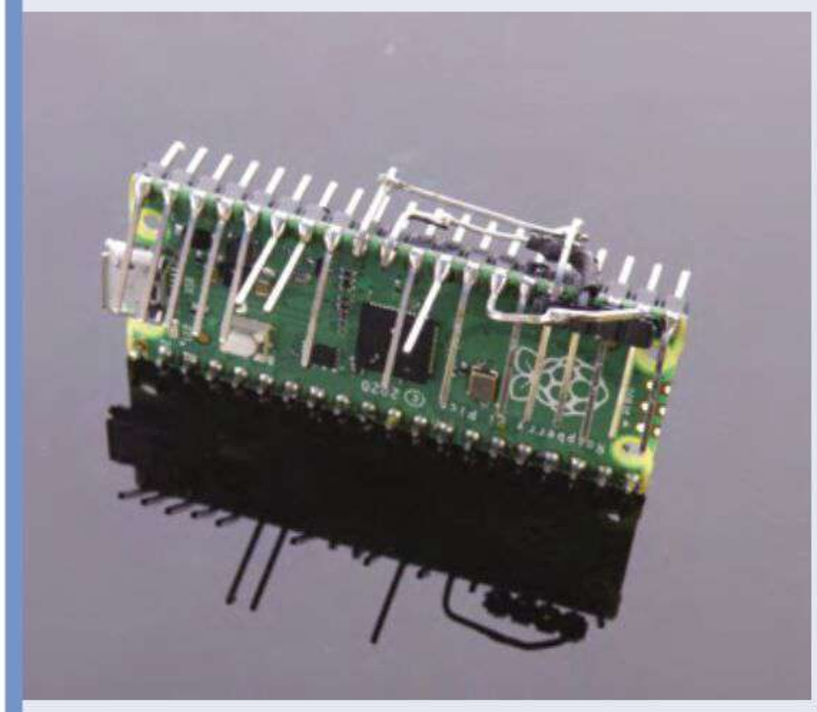


▼ Can you spot Pico?

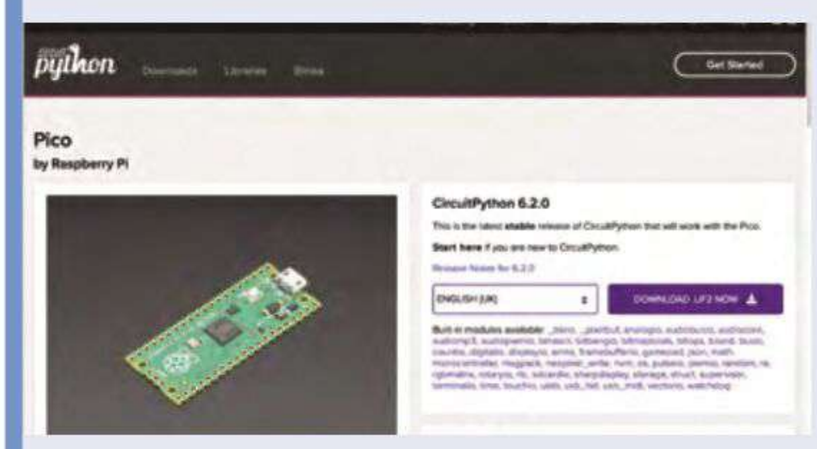


▼ A DS3231 real-time clock module provides accurate timing

2FA setup steps



01 You need a Raspberry Pi Pico with no headers yet attached, so that you can use double-sided, longer headers on one side. Solder the headers and plug the Pico into the keypad.



02 Install and update CircuitPython using the Raspberry Pico UF2 bootloader.

03 Copy the project source folder to the **CIRCUITPY** drive and adjust your **params.json** config file. See GitHub for format and details (**magpi.cc/picoth**).

Floppy Controller Board

Dr Scott M Baker is giving his old stash of floppy disks a new lease of life, as **David Crookes** discovers

MAKER

Dr Scott M Baker

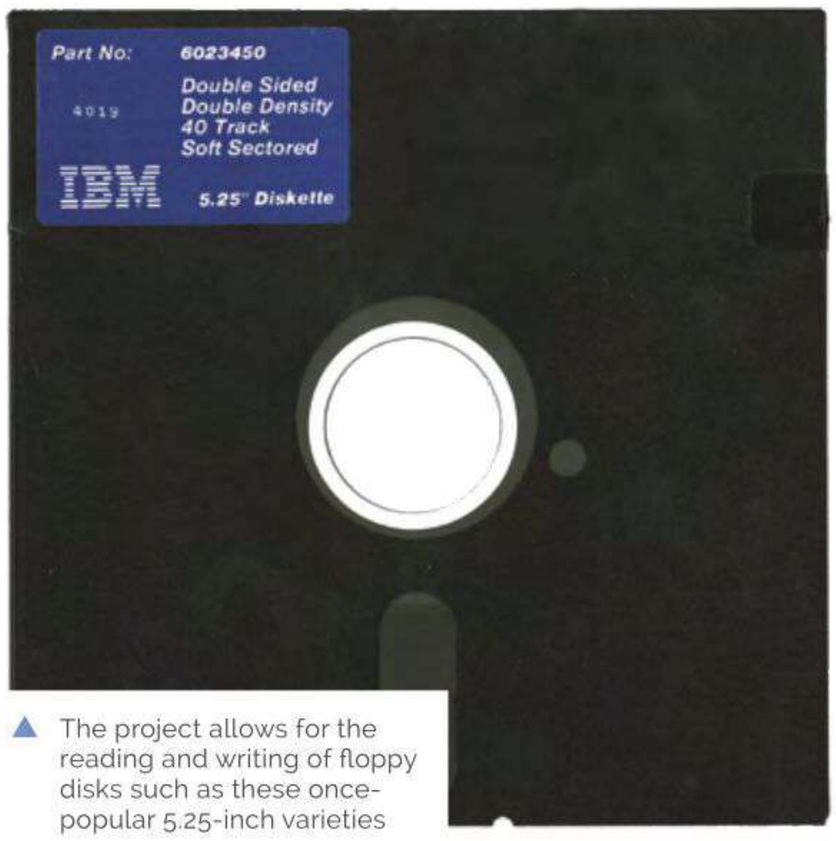
Scott has been interested in computing since the 1980s when he would write games on bulletin board systems. He is now an engineer working in the 5G networking space.

smbaker.com

Although computing is currently fixated with storing data in the cloud, there is still ample room for the use of physical media. Retro computer enthusiasts are certainly loath to let go of such age-old storage options, and that's why Dr Scott M Baker has decided to right one of the few wrongs of Raspberry Pi by allowing it to be hooked up to a floppy disk drive.

For the uninitiated, floppy disks were once cutting edge technology – so-called because the original 8-inch and 5.25-inch versions were thin and bendy (give one a shake and you'll see what we mean). They also came in smaller sizes including the more robust 3.5-inch and 3-inch varieties. Each offered fast and reliable storage – a world away from cassette tapes.

Scott feels nostalgic when he thinks of these disks, which is why he has created a floppy controller board. "The primary benefit is to make use of vintage hardware, in particular 5.25-inch disks which are not often usable on many newer computers," he says. "The device allows disk images to be read and written, making it possible to back up a 360kB floppy or create a floppy from a disk image to use in another vintage computer."



▲ The project allows for the reading and writing of floppy disks such as these once-popular 5.25-inch varieties

Time is of the essence

Creating the hardware proved to be fairly straightforward. Scott made use of the WD37C65 floppy controller integrated circuit – "a great single-chip solution," he says – because it combines the floppy controller, data separator, and control latch. It was used to create a HAT for Raspberry Pi which also has a 34-pin floppy header for the connection of a disk drive.

One of the key challenges came in creating the accompanying software. "It was tricky to get right due to the timing requirements involved," Scott explains. He started by taking the floppy driver from Wayne Warthen's ROMWBC project which allows an old disk operating system called CP/M to be ROM-based.

"It's written in Z80 assembly and I used that as a basis for writing a Raspberry Pi driver in Python with C extensions," Scott continues. "Since the software is written as a user-mode Python program, it may make it feasible to accommodate more unusual – that is, non-IBM PC – disk formats, or even reproduce vintage copy protection schemes."

Back to the future

To get to that point, however, Scott has had to overcome the fact that Raspberry Pi OS is not a real-time operating system. "That was the primary challenge," he says. "For high density drives, and by that I mean 1.44MB and 1.2MB, bytes must be read from the disk controller every 13 microseconds. For low density, they must be read every 26 microseconds.

"If Raspberry Pi is unable to read a byte in time, then the controller will declare a buffer overrun and abort the transfer. General-purpose Linux

Here, Scott Baker has connected Raspberry Pi to a Tandom TM100 5.25-inch floppy disk drive

Raspberry Pi has been placed into a Pi Camera Mount for stability, with the floppy controller drive HAT sitting on top

The cable for the floppy drive is placed into the 34-pin connector of the floppy controller board

Quick **FACTS**

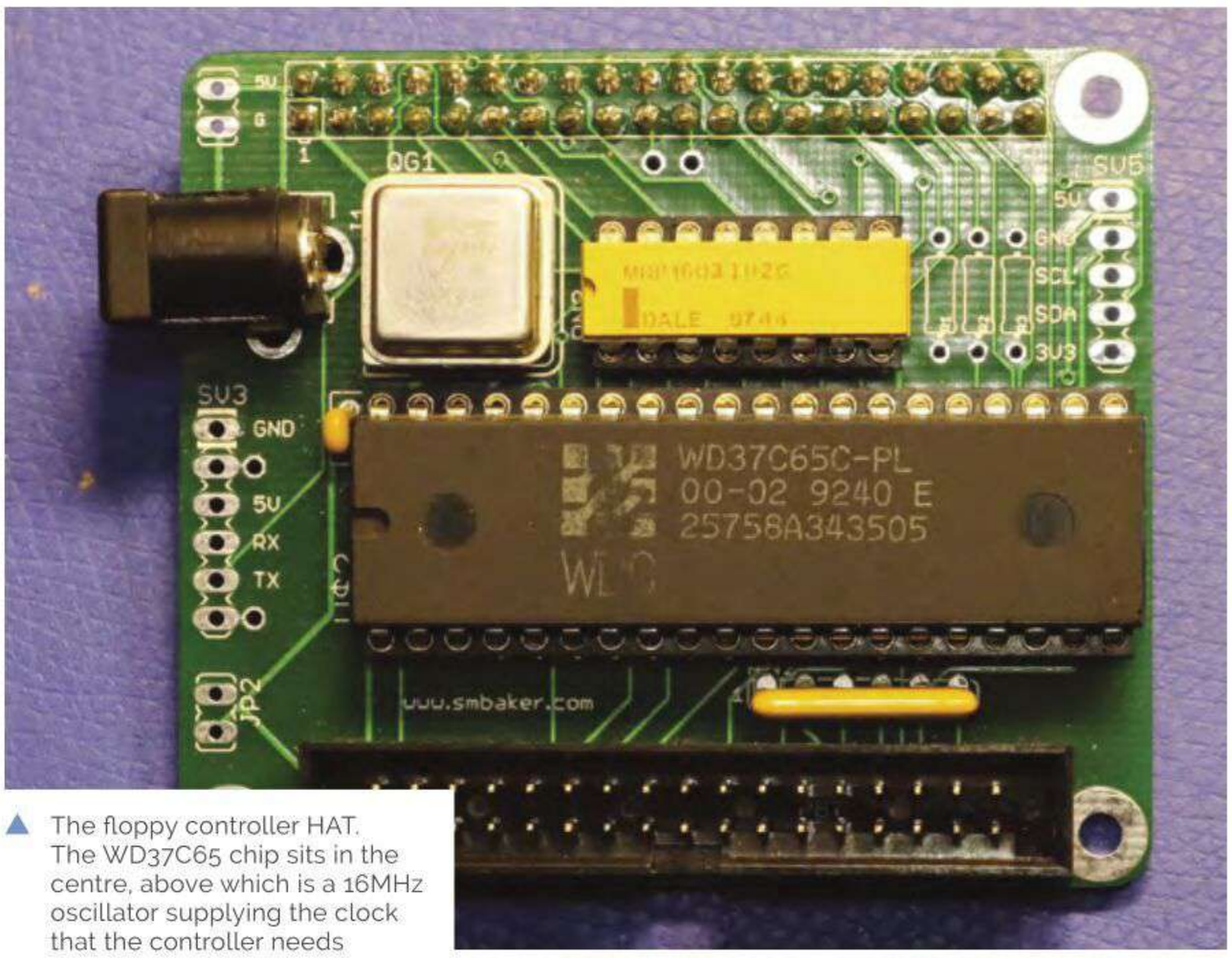
- The board is connected to a Raspberry Pi 4
- It will work with any floppy drive
- There's only one integrated circuit
- An optional barrel jack allows for 5V power
- A kernel driver would be the ultimate solution

> "The device allows disk images to be read and written"

does not provide the type of scheduling guarantees to user-mode processes necessary to meet these demands 100 percent of the time. The Linux kernel could decide to swap in another process at any time, leading to a multi-millisecond delay."

For help, Scott turned to the official Raspberry Pi forum and found that contributor 'tjrob' had worked on a technique to improve the real-time performance of the user-mode process. "Transfers rarely fail on low-density disks, but the failure rate on higher density disks may be around two to three per cent," Scott says.

Even so, the setup works and disks can be read and written using Raspberry Pi. It's not yet at the stage where you could fire up an emulator and run software from the floppy drive. "The driver doesn't provide the same interface as a Linux-block device, but that's something that could be added in a straightforward way," Scott says. Even so, it brings a new lease of life to an age-old format.



▲ The floppy controller HAT. The WD37C65 chip sits in the centre, above which is a 16MHz oscillator supplying the clock that the controller needs

Smart Buoy

Coastal change and rising sea levels are problems that are difficult and expensive to monitor. This exciting prototype could turn the tide. **Nicola King** dives in

MAKER

T3chFlicks

T3chFlicks is a small team that's on a mission to demystify the world of tech and make it accessible, fun, and relevant for everyone.

t3chflicks.org

When Steph, part of the T3chFlicks team, was writing her undergraduate dissertation on the impact of sea level rise in Grenada, she realised how difficult it was to get reliable, continuous data on the impact that sea levels were having on coastlines. "We wanted to create a solution," she tells us, "that would be cheap to build, easy to run, and provide meaningful data. In the bigger picture, it feels really important that we're able to build continuous datasets so we can get a more thorough picture of what's happening at our coasts."

It was also important for the team to try to understand the mechanisms driving coastal change. "Understanding how and why coasts are changing is the first step in robustly evidencing policy decisions to protect them," says Steph.

Buoy oh buoy!

After six months of planning and designing, including "two jam-packed weeks" in which the team physically built and deployed their prototype (see **magpi.cc/smartbuoy** for videos), Smart Buoy finally landed in Grenada's warm waters.

But, as Steph explains, there were some issues along the way. "The most challenging and time-consuming part was designing the Buoy casing and ensuring it was waterproof. While making the system solar-powered was pretty straightforward, ensuring it used minimal power required building a power scheduling system. We also set the bar pretty high in terms of what we wanted the Buoy to measure. Ensuring all the components worked to a good degree of accuracy was a massive challenge, and something we didn't quite achieve."

In addition, the team had problems with the memory of the Arduino used inside the Buoy, while SPI connections required a multiplexer. "During some initial tests, the sensors gave very odd results, so we had to go back to basics. On the plus side, the Raspberry Pi [used in the base station] worked flawlessly!" says Steph.

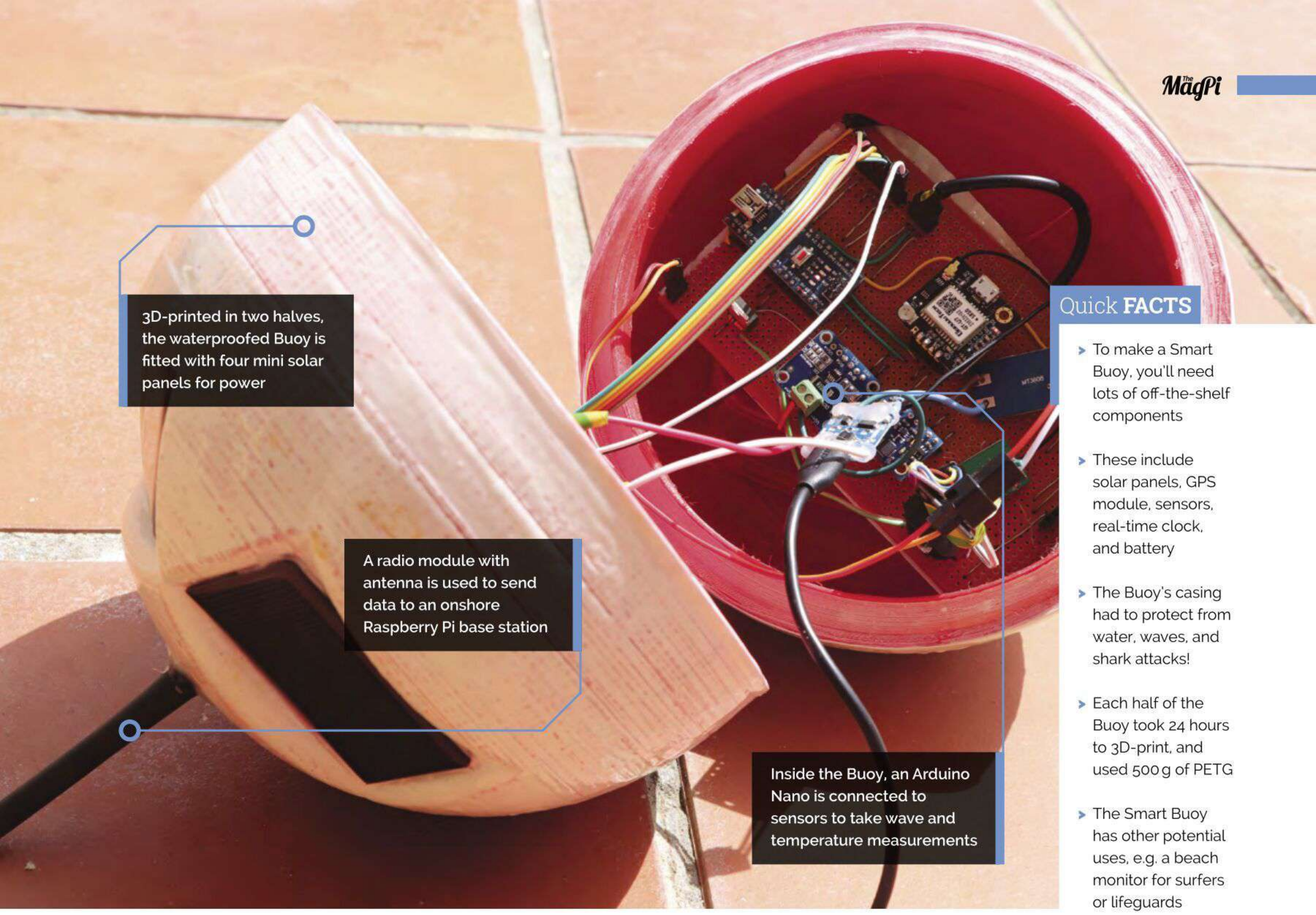
The Smart Buoy is essentially a solar-powered system of sensors which is controlled by an Arduino. "Once deployed, the Buoy sends measurements of environment characteristics (including wave height, wave power, and water temperature) via radio to a Raspberry Pi base station, which hosts a real-time dashboard and stores the measurements in a database."

The team chose Raspberry Pi as it's straightforward to program and connect to a radio module so it can communicate with the Arduino inside the Buoy. "Running Linux meant it was simple to run a database and a server for the dashboard. The maximum distance for the radio signal is 1km, so we had the base station on dry land while the Buoy was out at sea," explains Steph.

Hopes for the future

Although the team weren't able to keep the Smart Buoy in Grenada's waters for as long as they'd hoped, many positives came out of the exercise, including the lessons they learnt from the build. "We'd love to be able to invest more time in finessing the project and making more Buoys – a

▼ The Smart Buoy in the sea, taking wave and temperature measurements



Quick **FACTS**

- To make a Smart Buoy, you'll need lots of off-the-shelf components
- These include solar panels, GPS module, sensors, real-time clock, and battery
- The Buoy's casing had to protect from water, waves, and shark attacks!
- Each half of the Buoy took 24 hours to 3D-print, and used 500 g of PETG
- The Smart Buoy has other potential uses, e.g. a beach monitor for surfers or lifeguards

network around the island would be amazing, and would give you a really interesting, holistic (and hopefully meaningful) look at the coast."

In addition, they introduced their prototype to some officials from the National Science and Technology Council in Grenada: "Their feedback was that something like the Buoy, which gives reliable data with little manual input, would be invaluable." But, most importantly, Steph and the team have achieved their initial goal of making their Smart Buoy on a budget. "When we embarked on this project, it was to make a prototype to show that you could make something akin to a thousand-pound research buoy without splashing out."

While they know that Smart Buoy is still in prototype stage, the team hope to make improvements. "It would be amazing to see the Buoy in production, and we believe it would be a really valuable tool for education, citizen scientists, and even small governments."

Reaction from others has been positive too, as Steph reveals: "It made us so happy when a teacher in Spain got in touch to share that they were remaking the Buoy with their class... We really hope this project has inspired others, and would love to hear from anyone who could help us take the Buoy to the next level!"

> "It made us so happy when a teacher in Spain got in touch to share that they were remaking the Buoy with their class"


Testing the Buoy indoors. At the top left is the base station comprising a Raspberry Pi Zero with radio module

ShouldI: Renewable Forecast Display

Concerns about what powers our electronics prompted a Raspberry Pi project monitoring renewable energy use, reports **Rosie Hattersley**

MAKER

Andrew Brace

Full stack developer Andrew says Raspberry Pi's accessibility, compatible components, and community support helped him recognise the potential for his projects.

magpi.cc/shouldi

Andrew Brace built an energy tracker that can be used as a guide to scheduling household chores according to when they will use the least energy from fossil fuels – a neat way of addressing how to do his bit to tackle the climate emergency. Based on data from the National Grid, the renewables forecast display also makes use of publicly available information on energy pricing dependent on how much is being contributed by renewables. "The aim is to give people a quick, visual representation of the current status of the carbon intensity of our National Grid – that's how much of the electricity that's being generated is coming from renewable sources," Andrew explains.

"We all need to start thinking about how we can adjust our home activities to make use of times when renewable generation is high, and try to minimise high-power activities when the grid is using fossil fuels or other non-renewables," believes Andrew. A software developer by trade, he's used to using Python, PHP, and JavaScript for open-source projects, and has quite a number of personal Raspberry Pi projects under his belt.

Cooking without gas

For his renewable energy tracker, Andrew created Python scripts that fetch and parse data from the National Grid's Carbon Intensity API and from Octopus Energy's Agile pricing tariff API. "Octopus adjusts the price they charge customers for the electricity they use in 30-minute slots based on the renewable generation levels. Data about the pricing for the next 24 hours is published via a publicly accessible API," Andrew reveals. Most people will use it to calculate the cheapest times of day to run their washing machine or dishwasher, but this is an appealing, related option.

Andrew had already set up a service called ShouldIBake (**@baking4cast** on Twitter) that he created with two friends from UCL, with a similar aim of getting people to think about when they limit their energy costs and carbon fuel consumption.

▲ Using his ShouldIBake project, Andrew schedules sourdough loaf baking when the maximum amount of energy to heat the oven comes from renewables

For this project, he uses Raspberry Pi OS headless as no user interaction is needed. The details that a Raspberry Pi Zero W fetches are displayed on an Inky wHAT e-ink screen – itself a low-energy device. On-board wireless LAN means only one cable is needed to power everything.

There are four different layouts: the current level of renewable generation in your area; the forecasted levels for the next 4–5 days; a combination of those two; and Octopus Energy's current pricing tariff. You will be able to check the display and make a more informed decision about when to plug in, and also when it's best to unplug.

Andrew has given each information screen simple icons "that hopefully make it quick and easy to understand when renewable generation is high or low". His own display is currently hanging on his kitchen wall, just next to the door. "I see the data whenever I leave the room, so I always know when it's a bad or a good day to plug in."

A brighter outlook

Andrew plans to add a toggle button to the display itself to switch between screens. For now, he also

The graph shows energy consumption over the course of the day

This Raspberry Pi Zero-powered energy tracker is compact and light enough to fit inside a pine picture frame, then hung in a discreet corner

A Pimoroni Inky wHAT is ideal for displaying a mix of visual and numerical data

Quick **FACTS**

- The £70 project was built over the course of two weekends
- But it can easily be assembled in less than an hour
- This article about zero interaction graphics helped inspire the build: **magpi.cc/zerointeraction**
- It uses an Inky wHAT e-ink screen (**magpi.cc/inkywhat**)
- Another wall-mounted display shows information about his home environment

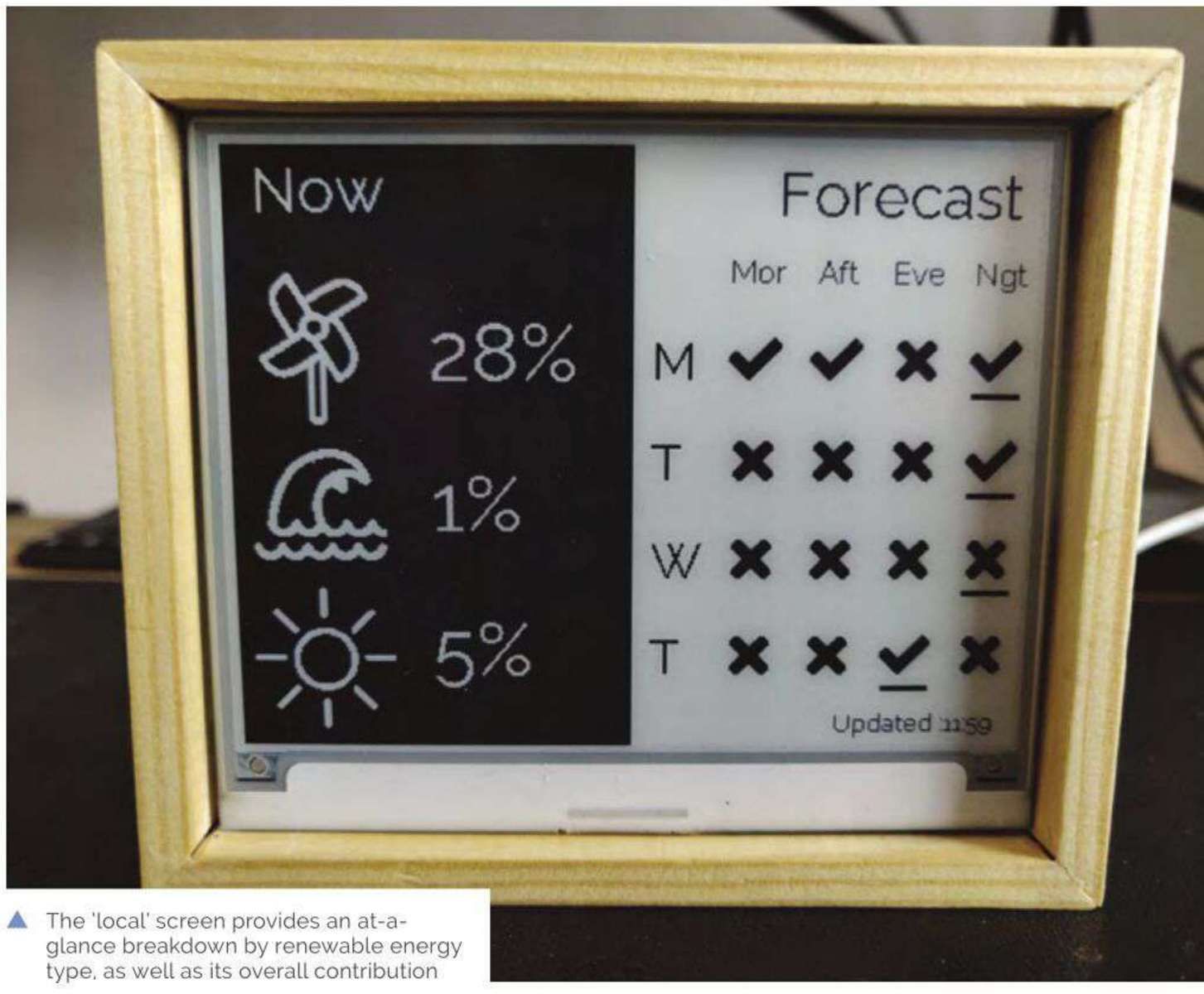
> Data about the pricing for the next 24 hours is published via a publicly accessible API

provides a simple web interface, made using the Flask web framework that allows for switching between the screens via a browser. Bookmarking this web page on his phone's home screen gives him quick access.

In time, he'd like to add a LiPo so the forecast data pulled from the Carbon Intensity API could be used to directly manage power consumption.

Andrew hopes that others will make use of his renewables forecast idea and plan their baking, hoovering, or charging their electric vehicles at the most environmentally appropriate time. "In this way," he suggests, "you can help balance the grid and the UK's transition to renewable energy."

True to his own principles, having converted a Citroen Relay van into an off-grid camper van complete with solar panels, he's in the process of adding Raspberry Pi energy consumption trackers to this too!



▲ The 'local' screen provides an at-a-glance breakdown by renewable energy type, as well as its overall contribution

Raspberry Pi Ri

Having trouble accessing the GPIO pins on your Raspberry Pi 400? **Rob Zwetsloot** discovers one young maker's solution

MAKER

Elijah Horland

A young maker and MythBuster, Elijah has always been a Raspberry Pi Kid, having his life changed after receiving one in 2015.

notabomb.org

While we love our Raspberry Pi 400, we find ourselves using a regular Raspberry Pi for electronics projects, so we're never too far away from the GPIO pins. Elijah Horland, a young maker who you may know from *MythBusters Jr.*, came up with a clever solution for just this purpose: "It's an 'underpinned' clip-on breadboard for Raspberry Pi 400, sort of a pin-pad for the keyboard form factor," says Elijah.

"I was using Raspberry Pi 400, and I had a small problem with the placement of the GPIO," he tells us. "I couldn't reach it without turning the entire thing, and it would stop my workflow to plug things in. The lack of labelling on a breadboard is also a problem for me. I keep GPIO pinouts up on my walls, but that's really just a patch, not a solution."

Elijah also managed to spread out the GPIO pins in the process, retaining some 'real-estate' on the breadboard.

Hidden benefits

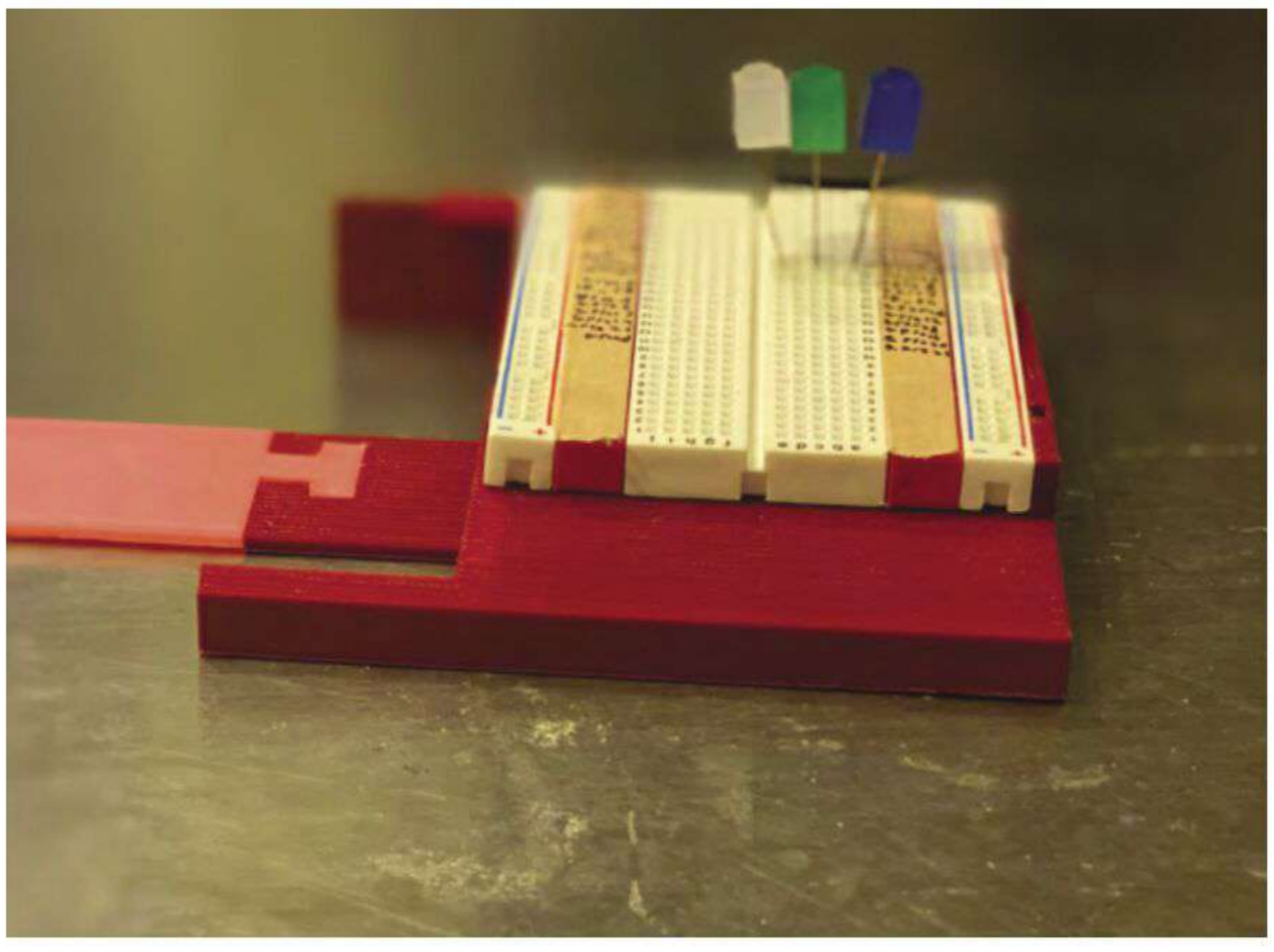
"While it does help me work faster, it would also help a beginner who does not know what GPIO pins are what, because it has a label on it," Elijah reckons. "On top of that, it hides the T-connector under the breadboard, which saves pins one-to-three on each side of the breadboard. When there are only five pins that directly connect with the GPIO pins, every saved space counts."

Apparently it also stops the whole thing sliding around a bit – the ribbon cable is stiff and has a habit of moving the breadboard around as it flexes and moves.

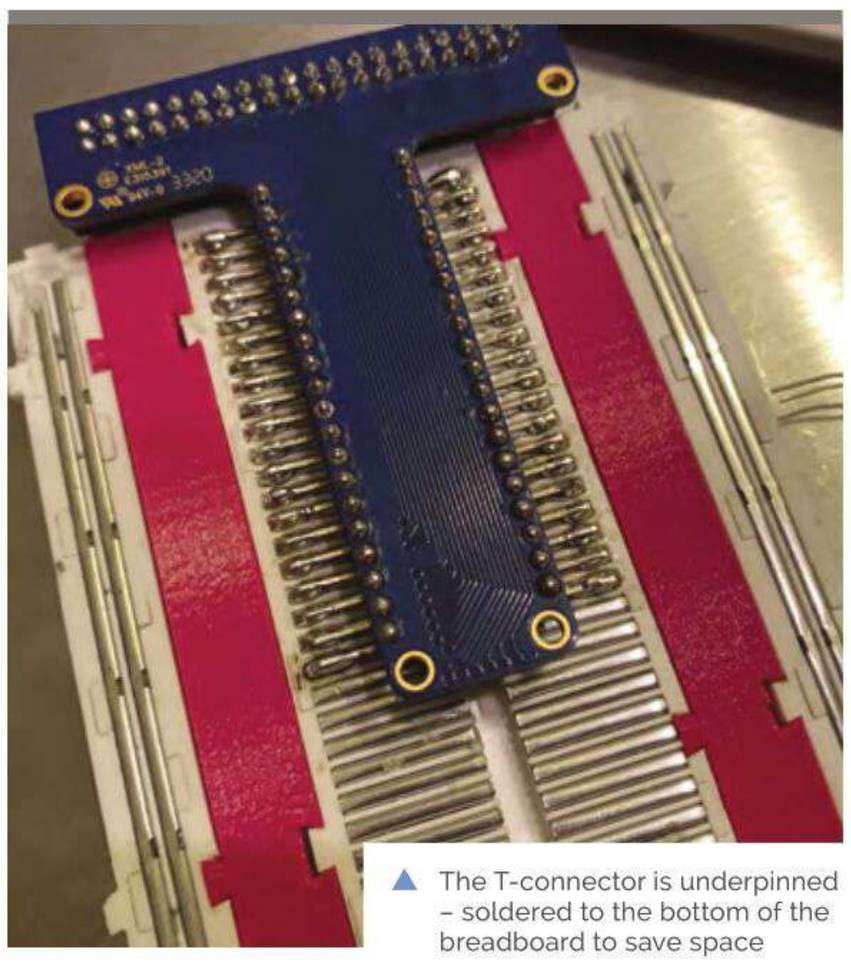
"The Ri fixes that by keeping the breadboard where a 'numpad' would usually be on a full-sized keyboard," Elijah explains. "If the normal breadboard size is too small for the project that you are doing, you can always just swap it out for a bigger breadboard, because it used the standard side pins of the breadboard to hold itself to the 3D print."

Now and then

While it's helped Elijah with productivity, he isn't finished with the project yet, and has some future plans for it.



▼ The 3D print is fairly simple, but it's optimised specifically for Raspberry Pi 400 for now



▲ The T-connector is underpinned – soldered to the bottom of the breadboard to save space

A ribbon cable connects the breadboard to the GPIO pins on the rear of Raspberry Pi 400

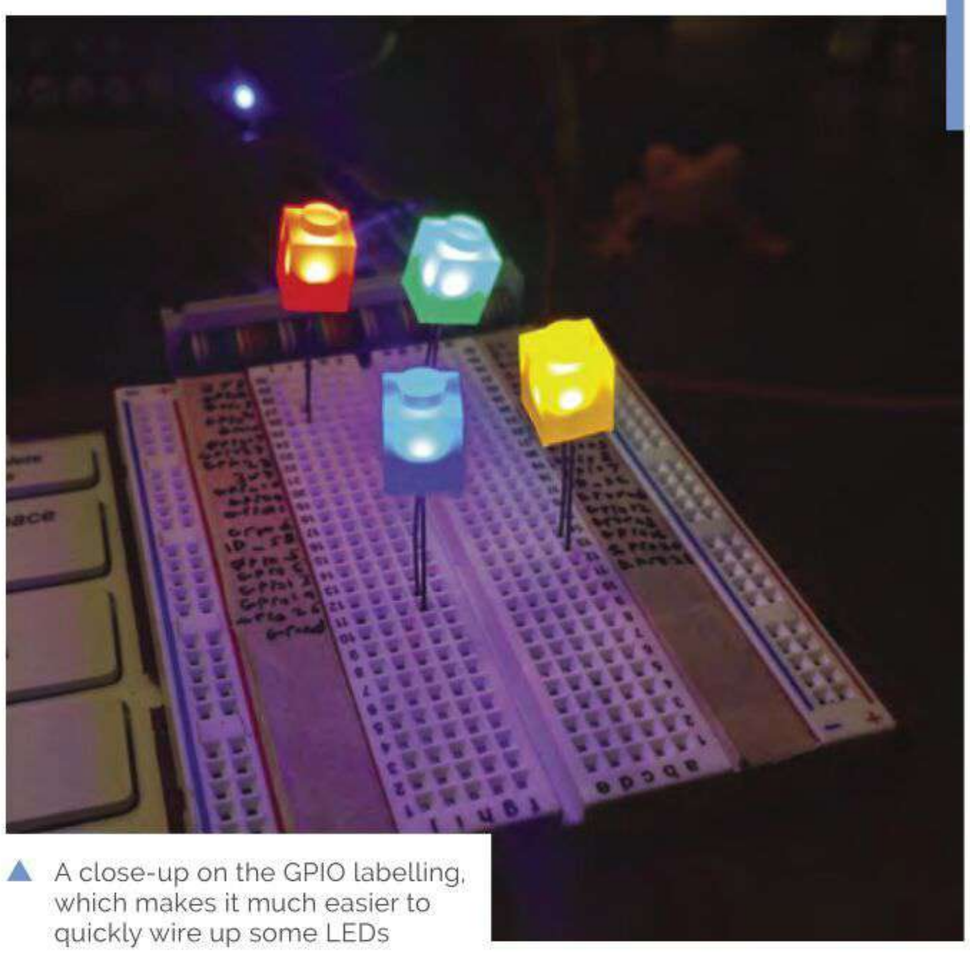
Ri is designed for the form factor of Raspberry Pi 400 – think of it as a numpad-like extension for electronics

Each pin is labelled, so there's no need to keep referring to a pinout guide

> I keep GPIO pinouts up on my walls, but that's really just a patch

"I plan to uncover the two vent holes by having the 3D print connect using the GPIO pins instead of clipping to the side of the board," Elijah says. "Underpinned trays for a breadboard for Pico, [Raspberry Pi Zero] W, and [Model] B form factors are also planned for after the school year."

If you want to make your own, Elijah has put the 3D model for the little underpinned holder up on Thingiverse here: **magpi.cc/rimodel**.



▲ A close-up on the GPIO labelling, which makes it much easier to quickly wire up some LEDs

Quick **FACTS**

- Raspberry Pi 400 is now Elijah's go-to Raspberry Pi
- The GPIO labels help him put together quick demonstrations
- Ribbon cables were popular GPIO breakouts in the early days of Raspberry Pi
- T connectors like the T-Cobbler can be found on **adafruit.com**
- The whole print is only three pieces

Drinks machine

MAKER

Spencer Organ

Spencer Organ is Head of Computer Science at a large school in Birmingham. He has been a member of the Raspberry Pi community since the early days and he likes to build fun, creative projects using the computer.

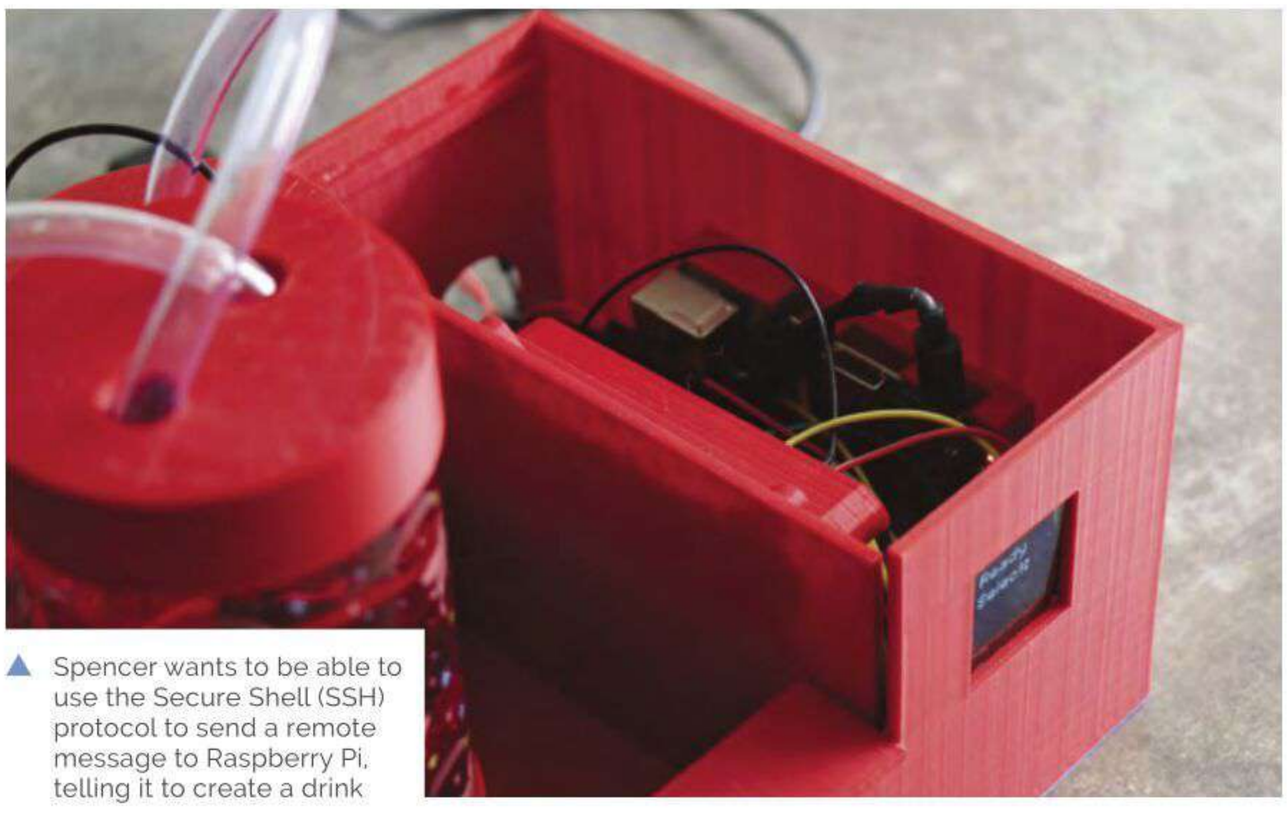
magpi.cc/makercupboard

Spencer Organ has been allowing his creative juices to flow by building a thirst-quenching automated drinks machine, as **David Crookes** discovers

Project making can be thirsty work, so Spencer Organ's drinks machine is certainly proving refreshing. All he needs to do is press a button and his device will pour him a fresh glass of squash. A screen will even tell him what drink is being dispensed.

"I'd been missing my meals in the local chain pub with their self-service drinks machines," Spencer says of his motivation for the project. "I wanted to make a machine that could sit on my desk and pour drinks while I was working."

At first, Spencer intended to create an automated home watering system. "Then, like many projects, things took a sudden twist," he laughs. His first big decision was whether to make use of a standard Raspberry Pi computer or Raspberry Pi Pico.



▲ Spencer wants to be able to use the Secure Shell (SSH) protocol to send a remote message to Raspberry Pi, telling it to create a drink

"Both would have worked well, taking inputs, displaying a message on the OLED display, and activating the relays," he explains. "In the end I went for a non-Pico Raspberry Pi computer as I liked the opportunity to SSH into it and select myself a drink from a different room (as long as the glass was under it)."

A glass act

The drinks machine incorporates a Touch pHAT, offering six touch-sensitive buttons. It also has an OLED display, a four-channel relay board, and two submersible water pumps. Spencer designed the case using Autodesk's Fusion 360 CAD software. "I printed it in a lovely red filament, opting for a modular approach with each piece slotting into each other," he says.

Although Spencer has created many projects (he can often be found at Raspberry Pi events up and down the country either speaking or helping out), for the drinks machine he insisted on paying great attention to the quality of the final build.

"This was the first project where I planned in advance the route that cables would take to make sure they were all hidden," he tells *The MagPi*. "This was also my first project where I used custom-made cable looms for 5 V, ground, and I2C with heat shrink to tidy them up. I didn't want it to look homemade, and I wanted it to dispense drinks without flooding my desk, too."

Going down well

Spencer has certainly succeeded in creating a robust, fab-looking project and he's been enjoying lots of cold beverages of late without ever needing

Raspberry Pi and the four-channel relay board are placed neatly in their own compartment that is then covered by a lid

The selection is made using the Touch pHAT panel located on the front of the device, with the choice showing on an OLED screen

Concentrated squash is contained in the two submersible water pumps at the back of the device, allowing for a selection of apple or lemon juice

Quick **FACTS**

- The device sits on Spencer's desk
- It mixes water with concentrated juice
- A Touch pHAT makes for a great controller
- It uses a solid 3D-printed case
- The whole device cost about £70 to make

Warning!
Food safety

The pumps are not sold as food-safe and care should be taken to keep the device clean and hygienic.

magpi.cc/foodhygiene

> I wanted it to dispense drinks without flooding my desk

to leave his chair. So how does it all work? "The drinks machine can either be operated by pressing the capacitive touch buttons or programmed to dispense a drink at a certain time," he says of his device's automatic credentials.

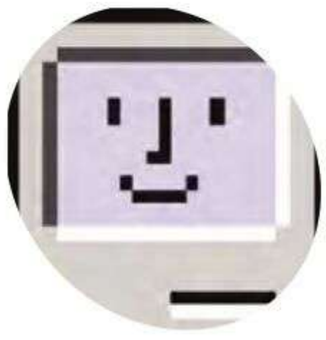
"When one of the capacitive buttons is pressed, a relay is activated and this switches on a small submersible pump." The drink is then dispensed into the glass, while the name of the drink is displayed. But there's room for improvement.

"I still need to add fail-safes such as empty glass detection," Spencer says, again looking to prevent liquid from spilling. "But this has been a fun build and I am looking forward to seeing what comes next with it."

▲ A drinks glass has filled up nicely, ready for Spencer to take a refreshing sip

Tiny Mac

One die-hard fan of the Classic Apple Mac has used Raspberry Pi to recreate a much-missed favourite. He tells **Rosie Hattersley** how



MAKER

Chuck Genco

Retired web developer Chuck first started mixing sculpture and tech in his free time as an antidote to a rather dull job. These days, he rather enjoys Raspberry Pi and Arduino projects

magpi.cc/makingtinymac

▲ Image credit: magpi.cc/happymac

Smart tech and tech built into sculptures sound quite innovative, but maker Chuck Genco says such ideas using relay logic were around in the late 1970s when he moved to New York and undertook an MFA in Sculpture. He completed his first microcontroller project 20 years ago, and has continued to incorporate technology into his sculpture designs ever since. His latest mixes 3D modelling and Raspberry Pi to create a Tiny Mac.

Mac envy

Tiny Mac is the first iteration of Chuck's endeavour to combine Raspberry Pi and sculpture. He first encountered Raspberry Pi when creating a programming prototype for a client and, with Raspberry Pi Zero, saw a chance to make his version of a 2013 shrunk-down Mac project he'd long admired (**magpi.cc/shrunkcomputer**).

"Tools, technology, and software have progressed since then, so I decided to make my own," says Chuck. "I owned a Mac Plus back in the day and I loved that computer," he adds.

Raspberry Pi Zero was the obvious choice because Chuck didn't want full-size USB ports sticking out of the side of the case. He chose the smallest possible display he could find, since this determined the scale of the case he'd need to design and 3D-print. Raspberry Pi Zero W was the best option, but meant missing out on decent sound output. A sound-board conflicted with the display, and Bluetooth provided only a partial solution but, having got some further ideas from other makers, Chuck is keen to explore some other sound options.

Sizeable issue

A bigger issue was the lack of room left over in the case once the 40-pin GPIO cable was inside. It took up most of the Tiny Mac's interior. Sound and space issues aside, Chuck says the whole project was surprisingly straightforward. He modestly describes his input as designing the case and "bringing together components and software that already exists". The process began with an ARM

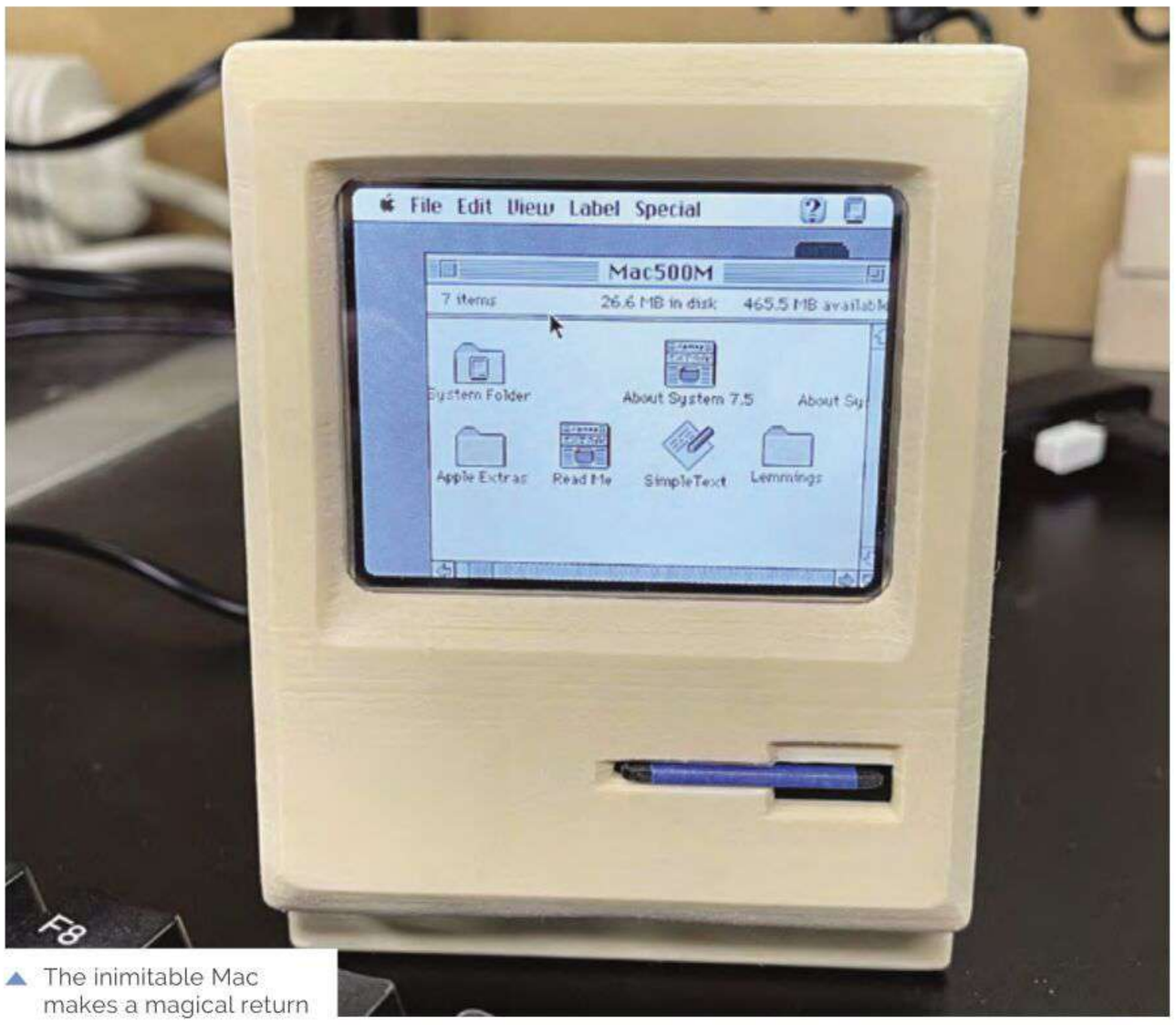
Chuck custom-designed the case and 3D-printed it using PLA filament as close in colour to Apple's original as possible

macOS software from The Gryphel Project Mac archive scripted to run on Raspberry Pi Zero W

A dummy floppy disk was created from plasterer's tape wrapped over a 3D-printed part

Quick **FACTS**

- Chuck is a self-taught programmer
- He learnt about microcontrollers, relay logic, and adjustable timers...
- ...during his ten-year stint fixing photocopiers
- The Tiny Mac cost around $90 to build
- His next build is a tiny Lisa/Mac XL



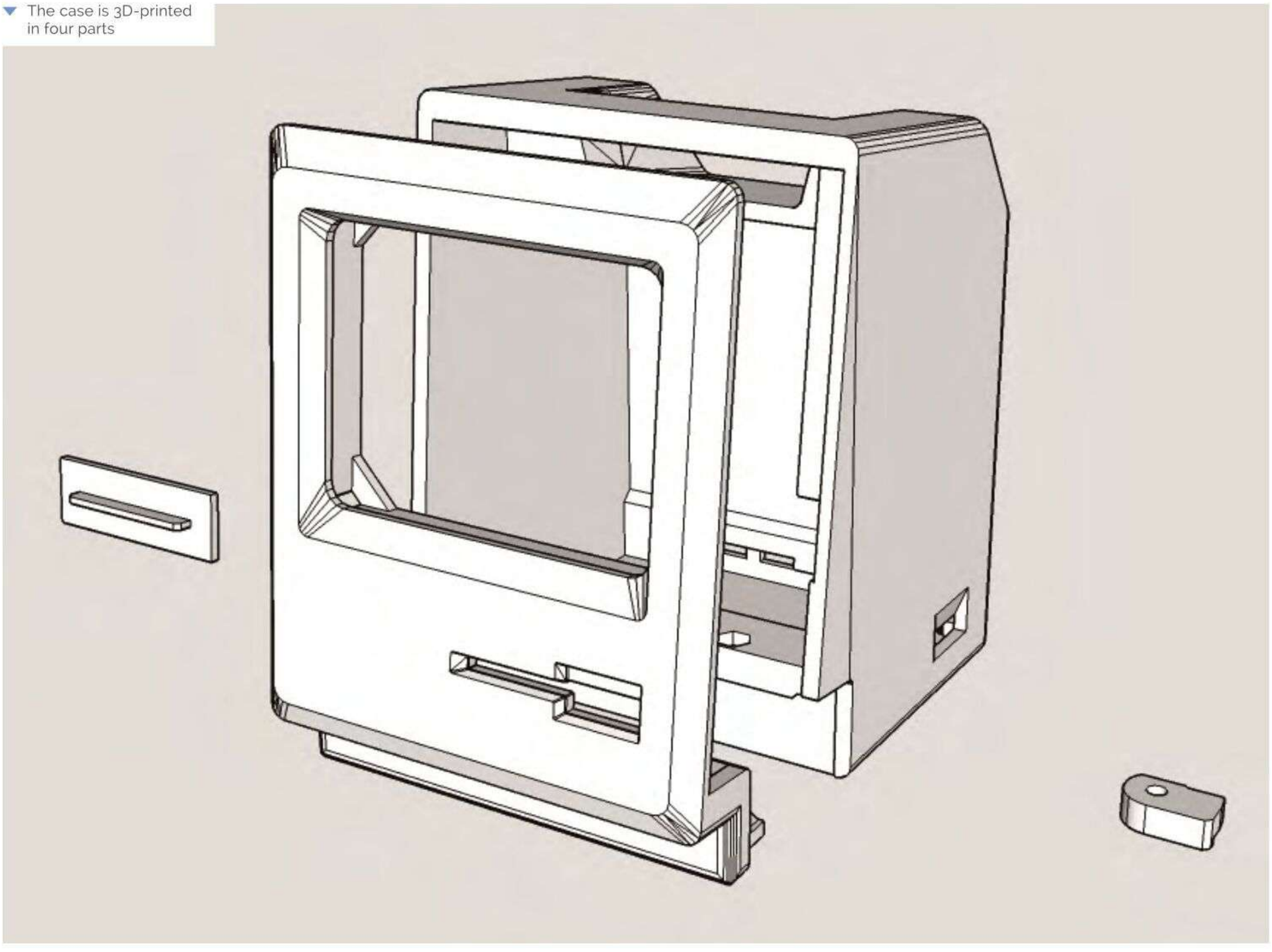
▲ The inimitable Mac makes a magical return

emulator on Chuck's Raspberry Pi 4, but it became a Raspberry Pi Zero project in order to avoid making a case with a wider bezel for the screen that he wanted. "There were many, many versions of the case," he says. "I can get very fussy about how things look." This extended to the 3D printer filament which needed to be the right Macintosh colour.

> "I can get very fussy about how things look"

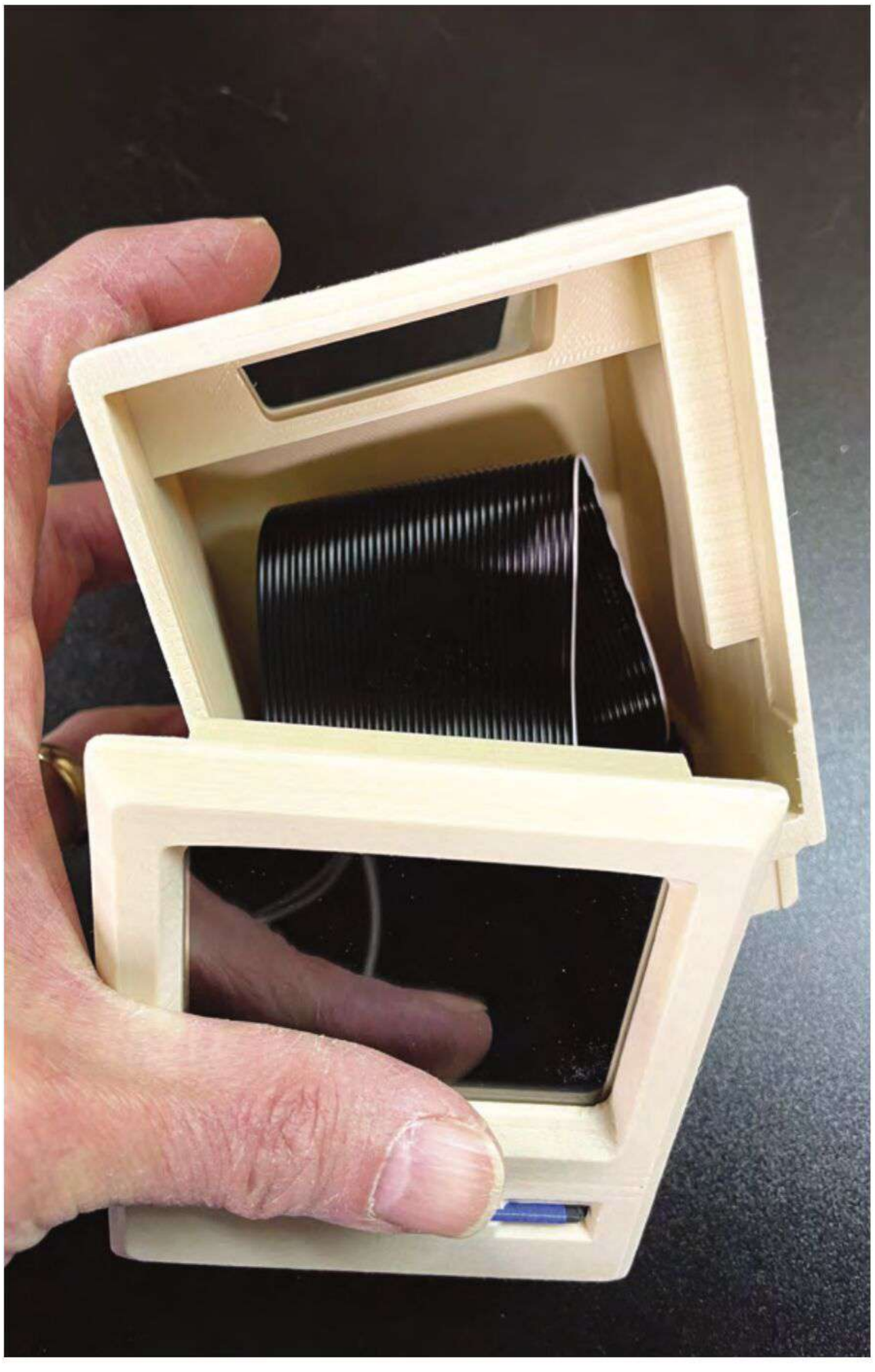
Having bought most of the hardware – Raspberry Pi Zero W, ribbon cable, display, 3mm hex screws and nuts, HDMI, microSD card, and power supply – online, he turned to the Gryphel Project (**gryphel.com**). The site exists to preserve early Macintosh computer software.

▼ The case is 3D-printed in four parts



Everything runs from Raspberry Pi OS desktop. Chuck wrote a startup script for the emulator but it wouldn't quit to the Raspberry Pi OS desktop, instead needing to be shut down via SSH. There are probably ways to work around this, he says, "but it was beyond my skills to make it happen."

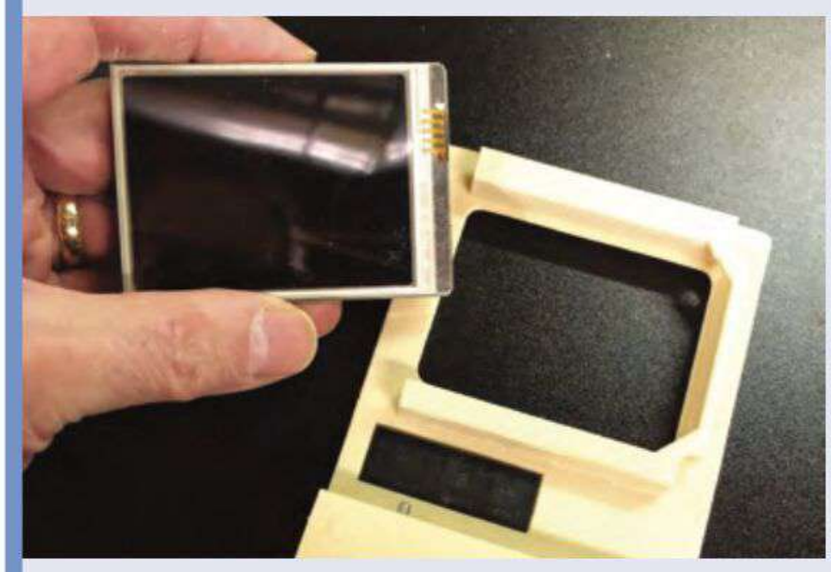
Chuck's instructions on building the tiny Mac also include links to other software that can be used on the retro computer. He also suggests using the Tiny Mac to run homebrew games or to host OctoPrint for 3D printing.



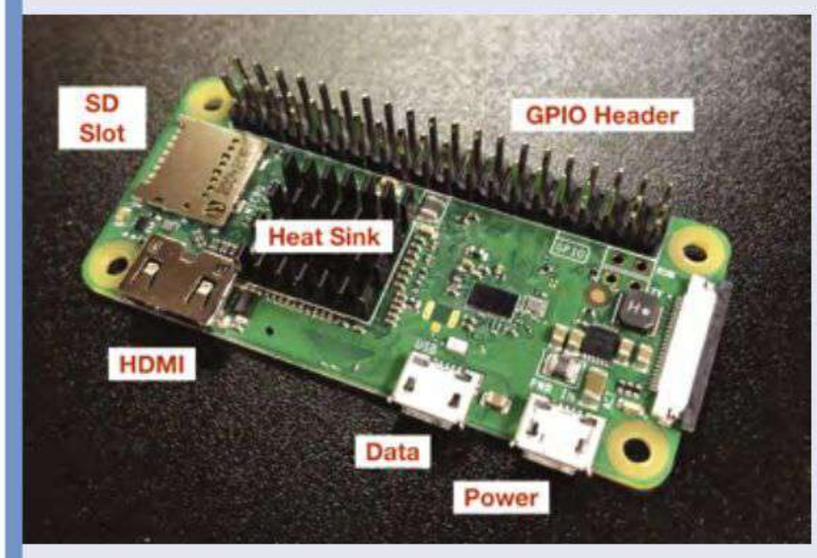
▼ You need a really short ribbon cable or you'll never fit it in your tiny Mac case

Make a Tiny Mac

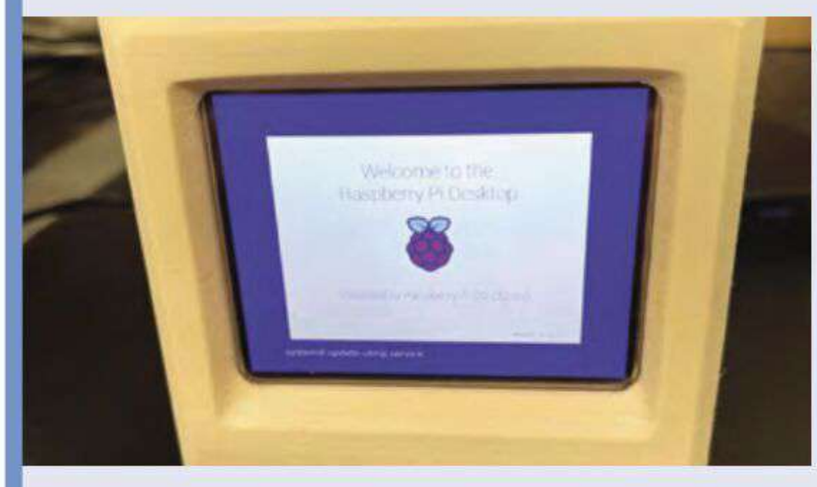
You'll need a suitably tiny screen and case for this project, 3D-printed or otherwise. Full case design templates and build instructions can be found at **magpi.cc/makingtinymac**.



01 Buy or 3D-print and build a case, and attach your 640×480-pixel display to the front. Insert Raspberry Pi's retainer toggle into the case's base.



02 Solder the GPIO header to Raspberry Pi Zero W (if needs be) and attach the heatsink. Download Raspberry Pi OS onto a microSD card, and attach the HDMI display, mini USB hub, keyboard, and mouse.



03 Power up Raspberry Pi, load the OS, and then enable SSH. Here it is shown running Raspberry Pi OS, but you can also use emulation software with Raspberry Pi.

Raspberry Pi Spigot

Here's an ingenious way of using a Raspberry Pi to calculate pi – and why not? **Nicola King** runs the numbers

MAKER

Adrian Chung

A software engineer who likes to dabble in hardware projects in his spare time. Interests include analogue circuits, microcontrollers, cardboard mechanisms, LED displays, and explaining mathematics using tactile or visual props.

magpi.cc/pispigot

Pi is an irrational number, which means it can't be expressed as the ratio of two integers. Since it has an infinite number of decimal places, calculating it to ever greater accuracy has long been an objective of mathematicians. So what better project for Pi Day (14 March) than to get a Raspberry Pi to calculate pi?

That's what Adrian Chung reckoned when looking to create a project for a 'speed round' contest. "I thought it would be neat to use a Raspberry Pi to compute pi to arbitrarily high precision," he tells us.

After looking into various methods, he learned about a class of algorithms that differed from the usual summing up of a sequence of decimal approximations. "Intriguingly, these so-called 'spigot algorithms' computed the next digit of pi after every few iterations of applying a small set of operations on a handful of integers," he notes.

Numbers on tap

Rather than simply using a Raspberry Pi to compute pi with a spigot algorithm, Adrian thought it required a more visual approach. "The need for a more visually explicit indication of what was actually running on the Raspberry Pi gave me the idea of creating a physical spigot prop with a tactile check valve that can be used to pause or resume the iterations of the algorithm," he explains. Upon the user turning the spigot, digits appear to flow from the tap and along an LED matrix display below.

"The MAX7219 8×8 LED display modules are daisy-chained SPI devices that are hooked up to the SPI interface on Raspberry Pi," says Adrian. "They are powered directly off the 5V rail; however, I had to add a separate power switch because they power up with all the LEDs turned on and this was pulling down the supply voltage during bootup."

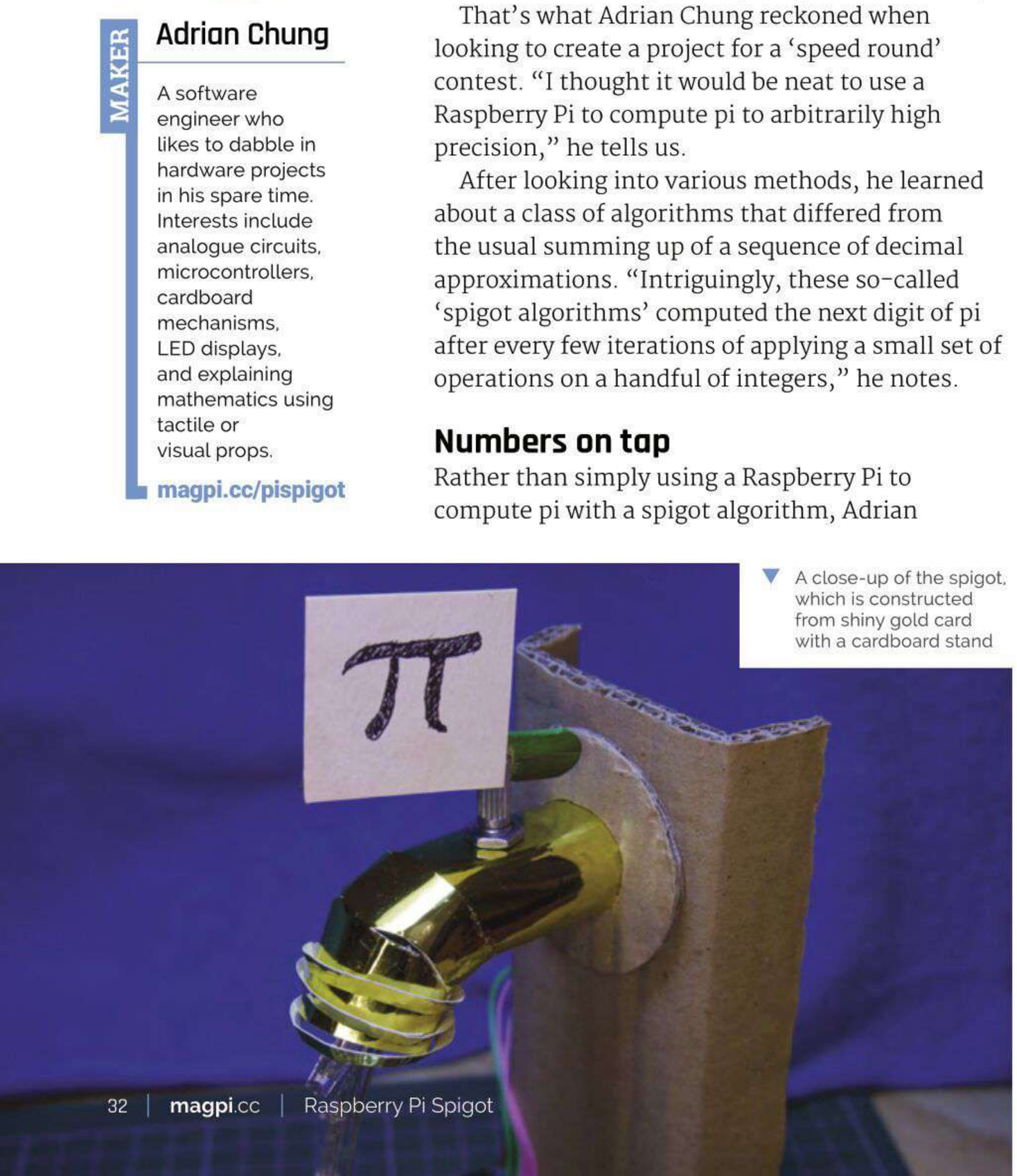
Three GPIO pins are used to animate the LED drips from the spigot. "The LEDs were cut from a Poundland Christmas decoration. Current is limited by 150 Ω resistors so that the drips don't appear overly bright against the scrolling display."

A potentiometer in the spigot is connected to two GPIO pins to check the valve position. "This works by using one pin to charge a capacitor through the potentiometer, forming an RC delay, and then timing how long until a logic high is read by the other pin."

Adjusting the flow

Adrian adapted an existing scrolling text demo script in the luma.led_matrix source code library: "I had to choreograph the dripping LED animation with the previous digits scrolling off to the left and the reveal of a new digit under the spigot."

He also needed to alter the potentiometer reading script, replacing the simple timing loop with regular system time queries for greater accuracy.



▼ A close-up of the spigot, which is constructed from shiny gold card with a cardboard stand

A potentiometer in the spigot is used to detect when it's turned by the user

An animation of three LEDs creates the illusion of digits flowing from the tap

The calculated digits of the number pi are shown on an LED matrix display

Quick **FACTS**

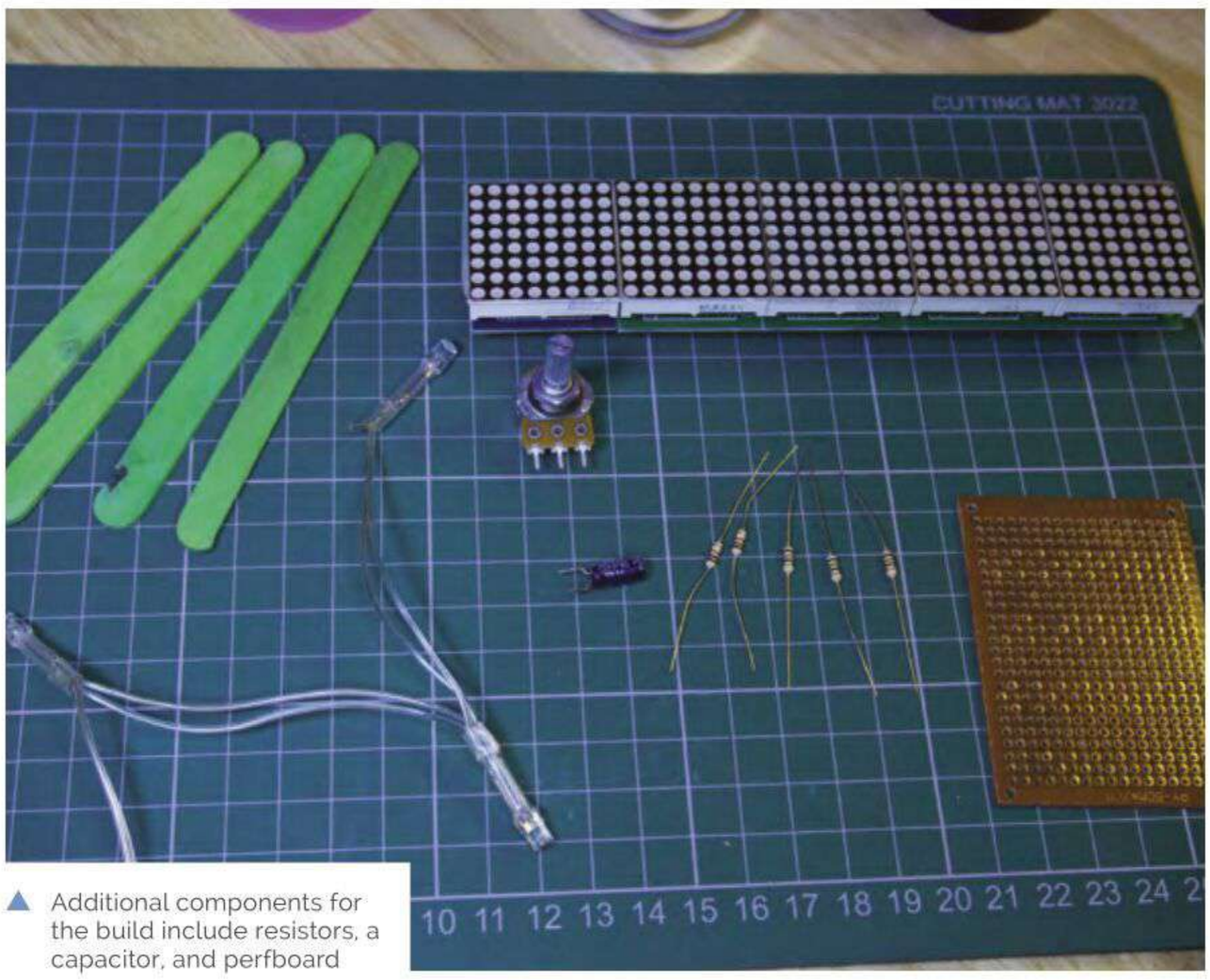
- Adrian spent five days researching algorithms for the project
- The actual build took him a couple of weeks
- This project won a prize in the Instructables speed competition
- Adrian's original tweet video of the project has over 10,000 views
- He'd like to explore using digits of pi to generate music or art

> The LEDs were cut from a Poundland Christmas decoration

So, how accurately can his Raspberry Pi Spigot calculate pi? "I left it to run for about six hours," says Adrian. "It computed more than the first 8000 digits. It can compute pi much faster than this, but the animation of the digits streaming from the spigot would just be a blur.

"Because those integer variables in the spigot algorithm only get larger, they continue to consume more and more RAM as more digits are cranked out. I don't really know how many digits of pi my 1GB Raspberry Pi 2 would have been able to calculate if I had just let it run."

Whatever the answer, the project has proved a hit with the community: Adrian's original tweet video has over 10,000 views and was retweeted over 100 times.



▲ Additional components for the build include resistors, a capacitor, and perfboard

SUBSCRIBE TODAY FROM ONLY £5

SAVE UP TO 35%

Subscriber Benefits

- **FREE Delivery**
  Get it fast and for FREE
- **Exclusive Offers**
  Great gifts, offers, and discounts
- **Great Savings**
  Save up to 35% compared to stores

Rolling Monthly Subscription

- Low monthly cost (from £5)
- Cancel at any time
- Free delivery to your door
- Available worldwide

Subscribe for 12 Months

£55 (UK) £90 (USA)
£80 (EU) £90 (Rest of World)

Free Raspberry Pi Zero W Kit with 12 Month upfront subscription only (no Raspberry Pi Zero W Kit with Rolling Monthly Subscription)

Subscribe by phone: **01293 312193**

Subscribe online: **magpi.cc/subscribe**

Email: **magpi@subscriptionhelpline.co.uk**

RASPBERRY PI BIG BUILDS

Be inspired to make something huge with **Rob Zwetsloot**

Tinkering with little breadboard circuits, programming games, or plugging it in behind your TV is a fantastic and simple way to use a Raspberry Pi. Sometimes though, we feel the need to get into something... Bigger.

We've been trying to make Big Builds a thing here in *The MagPi* for a while now, and as Eben is our witness, we'll make it happen.

So here are big, practical projects you can build using your Raspberry Pi. These are some of the most rewarding and satisfying things to make, especially when you can use the build every day, like many of the projects we've compiled here.

So grab your power tools. It's time to think big.

Warning!
Mains electricity & power tools

Be careful when handling projects with mains electricity. Insulate your cables and disconnect power before touching them. Also, be careful when using power tools during these builds.

magpi.cc/drillsafety
magpi.cc/electricalsafety

MAGIC MIRROR

A classic project for Raspberry Pi makers

Also known as a smart mirror, these easier-than-you-think big builds have become a big craze in the maker and Raspberry Pi communities. Got a spare TV? Got a wood-saw and a hammer? Then you're half-way to having your own *Iron Man* fantasy.

CHECK OUT ISSUE 90 FOR MORE magpi.cc/90

URL: MAGICMIRROR.BUILDERS
SKILLS: BASIC CARPENTRY
LEVEL: INTERMEDIATE

01 Build your mirror frame

If you have the right size of display, you may be lucky and find a frame at a hardware or furniture store that will fit. Otherwise, you may need to get some planks of wood and wood for the fascia to contain it all. Remember: measure twice, cut once.

MATERIALS

- Flatscreen TV or monitor
- Two-way mirror or mirror film and glass
- Wooden frame or suitable wood to create a frame
- Raspberry Pi

02 Set up software

Install a fresh version of Raspberry Pi OS onto an SD card and go through the initial setup. Make sure the wireless network it's connected to works where you plan to place the mirror. Once that's all done, open the Terminal and type: `bash -c "$(curl -sL https://magpi.cc/MirrorInstall)"`

03 Install your hardware

With Raspberry Pi all set up, all you need to do is plug it into your mirror display and place it within the frame, sandwiching the mirror/glass between the display and the frame. You can secure it with brackets or glue, depending on whether you'd like to remove it later, and then hang it in its home.

ARCADE MACHINE

CHECK OUT ISSUE 63 FOR MORE magpi.cc/63

Don't pay over the odds for just one game

We've all dreamed of having our favourite arcade game in our home. Arcade machine ownership has its up and downs, though, especially when it comes to maintenance. This can easily be solved by building your own custom cabinet that can play all your favourites.

URL: MAGPI.CC/ARCADECABBUILD

SKILLS: CARPENTRY, ELECTRONICS, GRAPHIC DESIGN

LEVEL: EXPERT

MATERIALS

- 48-inch (122 cm) piano hinge
- 24-inch (61 cm) soft-close drawer slides
- 2 × ½ inch (12.7 mm) overlay face frame concealed hinge (optional)
- Magnetic catch (optional)
- ¾ inch (19 mm) T-Molding (optional)
- Monitor
- Speakers
- Speaker grilles
- LED arcade buttons
- Joystick
- I-PAC or Pimoroni X-HAT
- Switches and assorted wires

01 Get building

This build is not simple – there are 24 steps in the original feature for this! Bob Clagett, who built this incredible arcade cabinet, has digital plans for the carpentry part here: **magpi.cc/2yboyPp**. You'll need a decent set of tools, or access to a well-stocked makerspace!

02 Install the electronics

Since we wrote about this arcade build, sourcing parts and connecting them to Raspberry Pi is now easier than ever. If you've followed Bob's guide, you'll have pre-cut holes ready to put in all your parts, and you'll be able to wire them up to a board to connect to your Raspberry Pi with little effort. Be careful with lights and speakers, though.

03 Set up Raspberry Pi

The simplest method in our mind is to use RetroPie on whatever Raspberry Pi you choose. With fantastic controller configuration settings for even the most unique layouts, and powerful emulation tools, this should be the easiest part of the build.

PINBALL MACHINE

Become a pinball wizard with this massive upcycle

Classic pinball machines are a nightmare to maintain, and take up even more space than arcade machines. Building your own custom one means you can use modern, low-power, and easily fixable tech. Just make sure you have a spare toddler bed lying around.

CHECK OUT ISSUE 78 FOR MORE magpi.cc/78

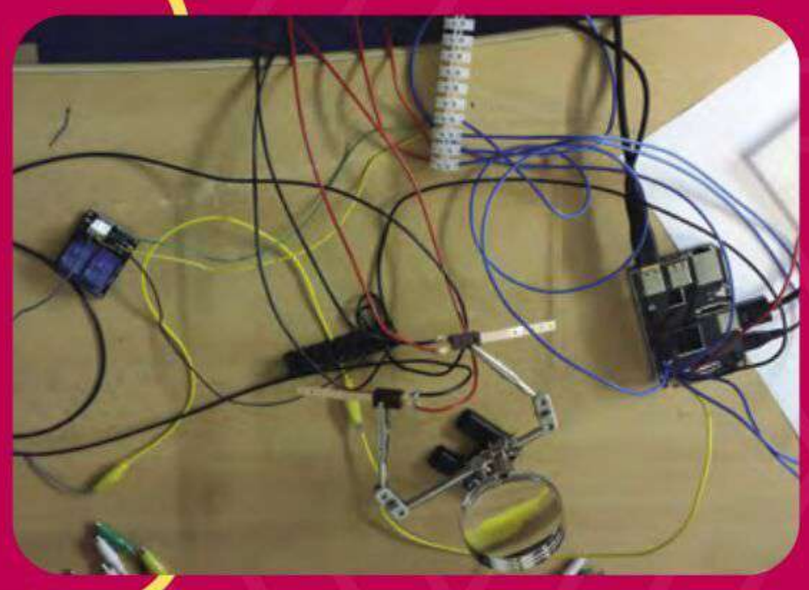
URL: MAGPI.CC/78

SKILLS: WOOD WORKING, ELECTRONICS, ENGINEERING

LEVEL: ADVANCED

01 Wire it

The wiring in a pinball machine is quite complex – there's a reason five metres of wire is recommended for it. Be careful as you wire it up, and make sure to test it first as well; you don't want to have to take it all apart to fix stuff later on.



02 Plan it

All your gizmos and bumpers and noise makers wired up? Time to plan their exact placement. Lay the parts out carefully and make notes, pencil in locations, and double-check everything. You could even figure out a scoring system now as well.

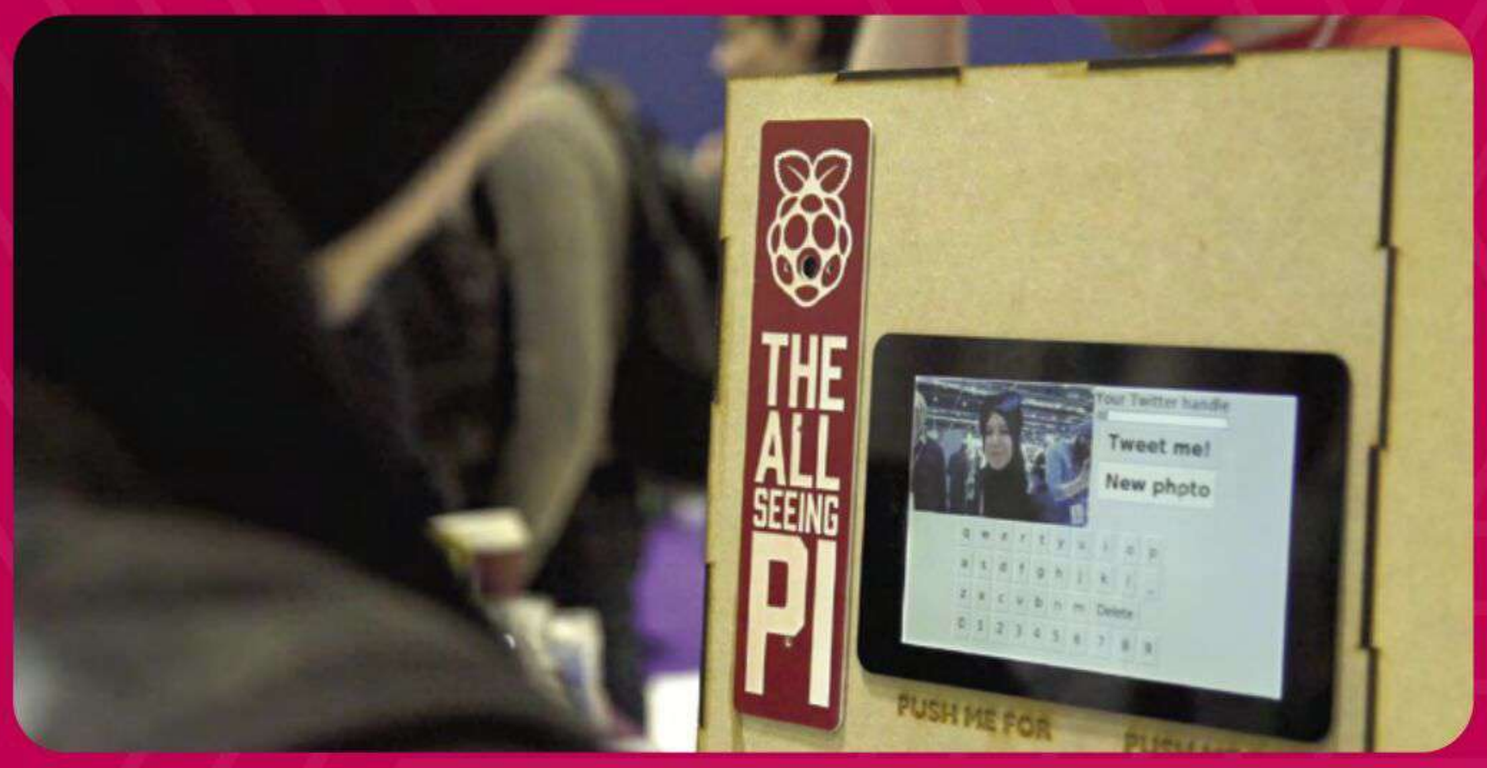
03 Build it

With everything laid out and marked, it's time to do the proper building. Get your drill and safety gear out and start cutting holes. Carefully install the components, lights, and other electronics (including a Raspberry Pi). This may take some time, so don't get too frustrated!

MATERIALS

- Old toddler bed
- Pinballs
- Pinball rubber rings kit
- Shooter assembly
- Cabinet flipper button nut
- Cabinet flipper button
- Slingshot cover (right)
- Slingshot cover (left)
- Flipper button switch
- Rollover microswitches
- Microswitch
- Spinner assembly
- Full flipper assembly with 1 coil FI-11630, normally open EOS switch (right and left)
- Star posts
- Pop bumpers
- 100 pc heat-shrink tubing
- 5 m 0.75 mm² multicoloured electrical wire
- 36 V power supply (TDK-Lamda LS150-36)
- 20 cm male-female jumper cables × 40
- 2 m LED light strip
- 5 V power supply (TOOGOO(R) AC 110 V/220 V)
- SainSmart 8-channel 5 V solid-state relay module
- SainSmart 2-channel DC-DC 5V-220V 5A solid-state relay

PHOTO BOOTH



Add extra fun to parties with the All-Seeing Pi

This Raspberry Pi Foundation project is something you can both see at official Raspberry Pi events and the Foundation offices, and build using their handy project resource. Photo booths are a great addition to a party, and this one allows you to add props as stickers and tweet them out as well!

URL: MAGPI.CC/ALLSEEINGPI

SKILLS: ELECTRONICS, PROGRAMMING, CARDBOARD FOLDING, OR CARPENTRY

LEVEL: NOVICE

01 Make a case

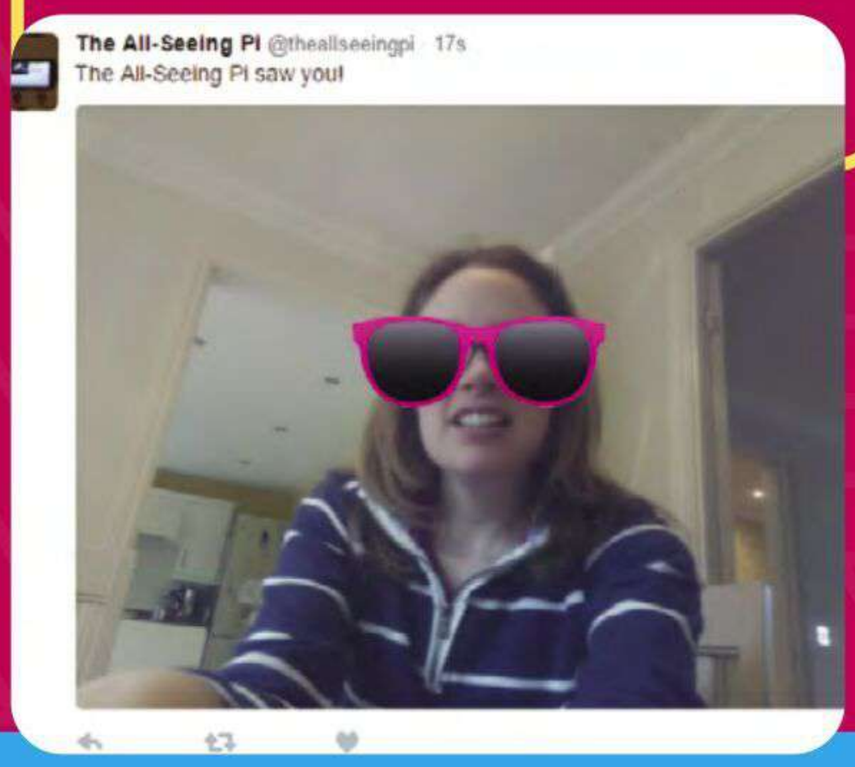
The first version of All-Seeing Pi was made out of a cardboard box. It's a great way to recycle delivery boxes and it can also be used to take measurements for creating a wooden version. The one you'll see in real life uses laser-cut wood.

MATERIALS

- Raspberry Pi Camera Module
- Raspberry Pi touchscreen display or standard monitor
- Tactile push buttons
- Breadboard
- Male-female jumper leads
- Large buttons (optional, to replace tactile push buttons)
- Cardboard or wood for the case

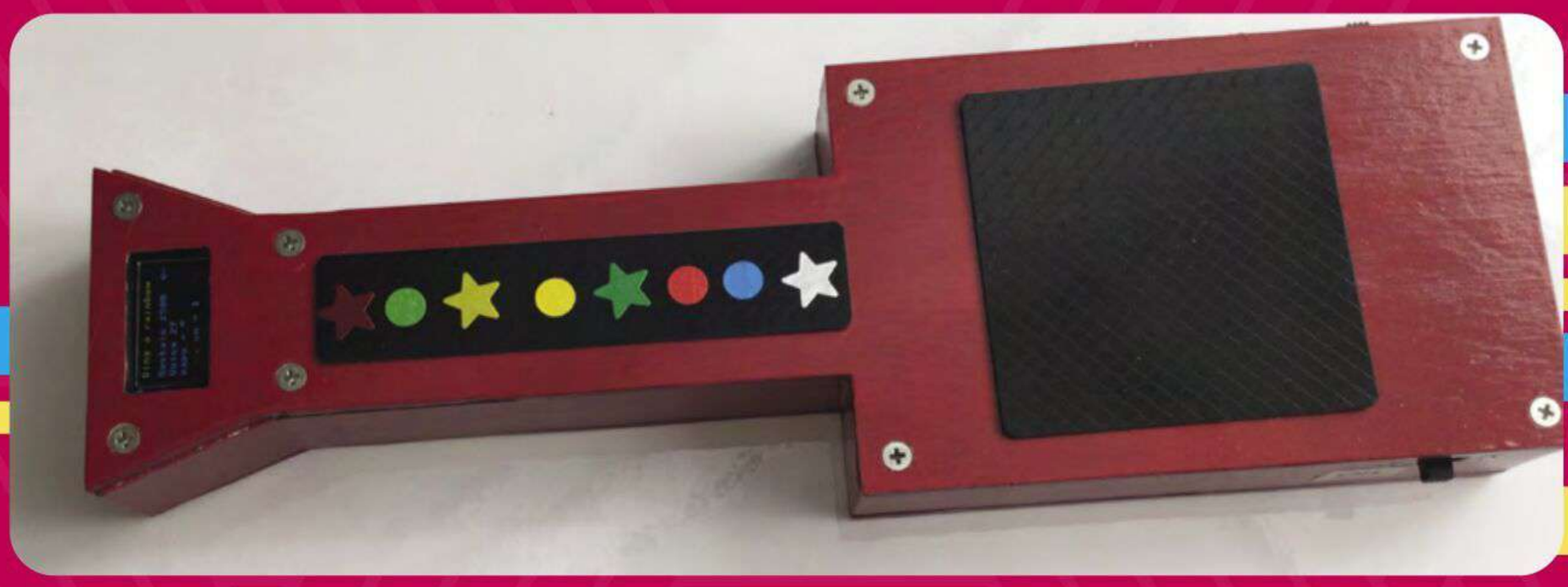
02 Program Raspberry Pi

The code for this project allows for several things: taking a photo, editing it, and then tweeting it. Each part uses separate libraries and functions, but can easily be combined using the tutorial in the project resource.



03 Install the components

With your cardboard box or wooden frame ready, and Raspberry Pi programmed, you can do a final test of the electronics before assembling them all, carefully, into the case. Maybe include some paper instructions for people who are a bit technophobic as well.



MIDI GUITAR

URL: MAGPI.CC/PIBAKERY

SKILLS: CARPENTRY, ELECTRONICS, PROGRAMMING, MUSIC PLAYING

LEVEL: EXPERT

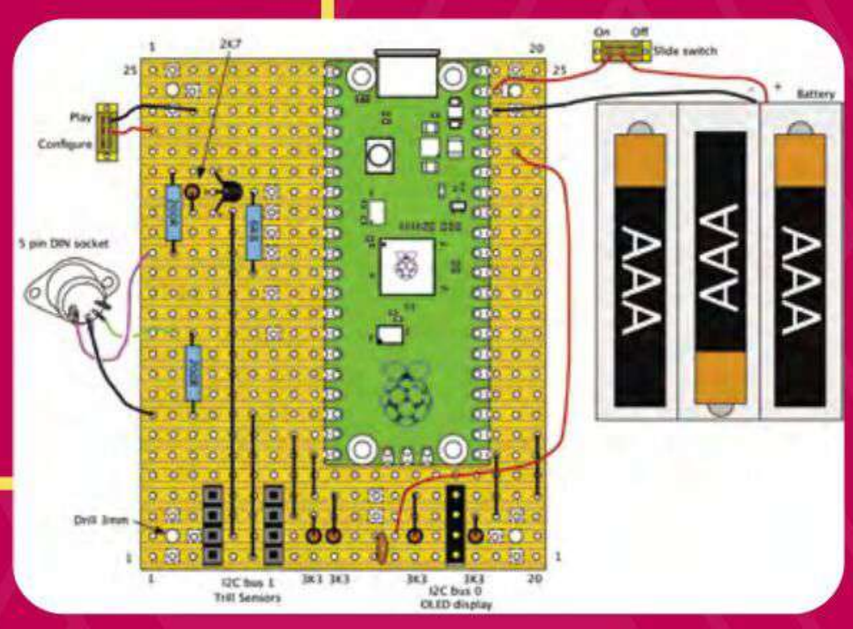
CHECK OUT ISSUE 104 FOR MORE magpi.cc/104

Play sweet licks with this fully electronic guitar

From issues 102 to 105, Mike Cook has been teaching us how to use Trill touch sensors with Raspberry Pi, and last issue he finished off with an amazing custom MIDI Guitar that you can build and play. We love it so much that we want to remind you of its existence and highly recommend you consider building it!

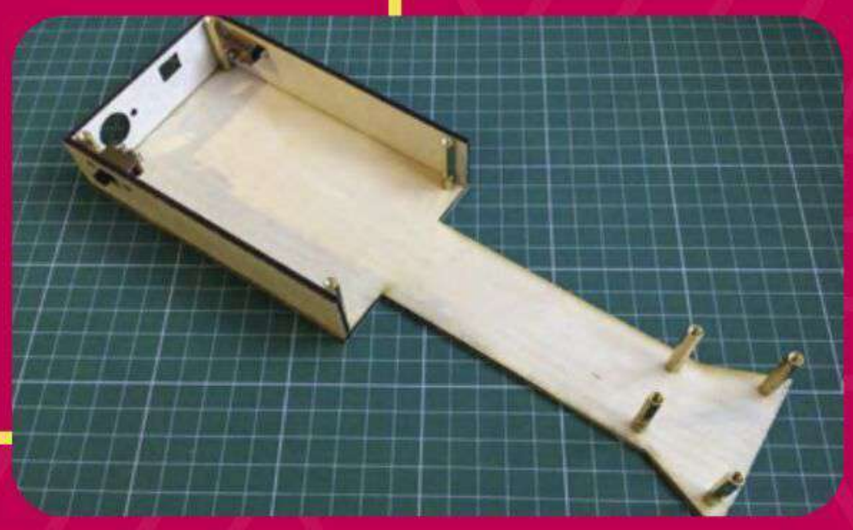
01 Build your circuits

For this guitar, Mike built a custom circuit board using stripboard, a sort of step up from breadboards. A Pico is mounted to the board and used for running the display. The Trill sensors attach to the board via connectors so they can then be read by Pico.



02 Build a guitar

A fairly simple guitar shell is made out of 3 mm plywood, at roughly the size of a ukulele. Make sure there's plenty of space for the electronics to go inside before it's fastened. We like the scarlet-red paint job Mike gave his version, but you can paint it however you wish.



MATERIALS

- Trill Bar and Square sensors – **magpi.cc/trill**
- OLED 128×64 I2C display SSD 1306 yellow/blue
- Raspberry Pi Pico
- 3 mm plywood
- Stripboard
- Various components
- MIDI sound generator

03 Tune it

It's not really being tuned like a traditional string instrument, more calibrated using cool stickers to know where the 'frets' are to play different chords. Mike fully explains many ways to program it, from free-play to having specific songs programmed into it.

Part 03

Build an arcade machine: Command and control

We've assembled our cabinet. Now it's time to put Raspberry Pi to work with the Recalbox arcade OS

MAKER

K.G. Orphanides

K.G. is a writer, maker of odd games, and software preservation enthusiast. Their family fully supports the idea of an arcade machine in the living room.

@KGOrphanides

With our arcade cabinet built, it's finally time to get emulating with Raspberry Pi. We're using the Recalbox emulation distro for this project, which has excellent GPIO arcade controller support, a slick front end, and a handy web interface to help you configure and manage it.

The RetroPie distro is a popular choice for arcade machines, and adds Steam Link support, but requires manual installation and pull-up switch reconfiguration to get GPIO arcade controls working.

01 Wire up your controls

Last month, when we added buttons to our cabinet, we recommended attaching the spade-to-DuPont cables that will connect to Raspberry Pi's GPIO before inserting the buttons. If you didn't, it's time to open the back of your cab, grab a torch, and get in there to fit them.

Connect a spade-to-DuPont cable to each button and connect a shared ground cable to each of the left and right button banks. Where you have longer stretches of buttons – for example between the central hot button connected to player one's rig and the player one start button – it's a good idea to skip a connector on the ground chain to give yourself some extra cable to play with.

Plug the 5-pin cable into the joystick. Looking at our Sanwa stick from below, the bottom-most pin, which connects to the black cable strand on standard-coloured 5-pin wiring harness, is ground.

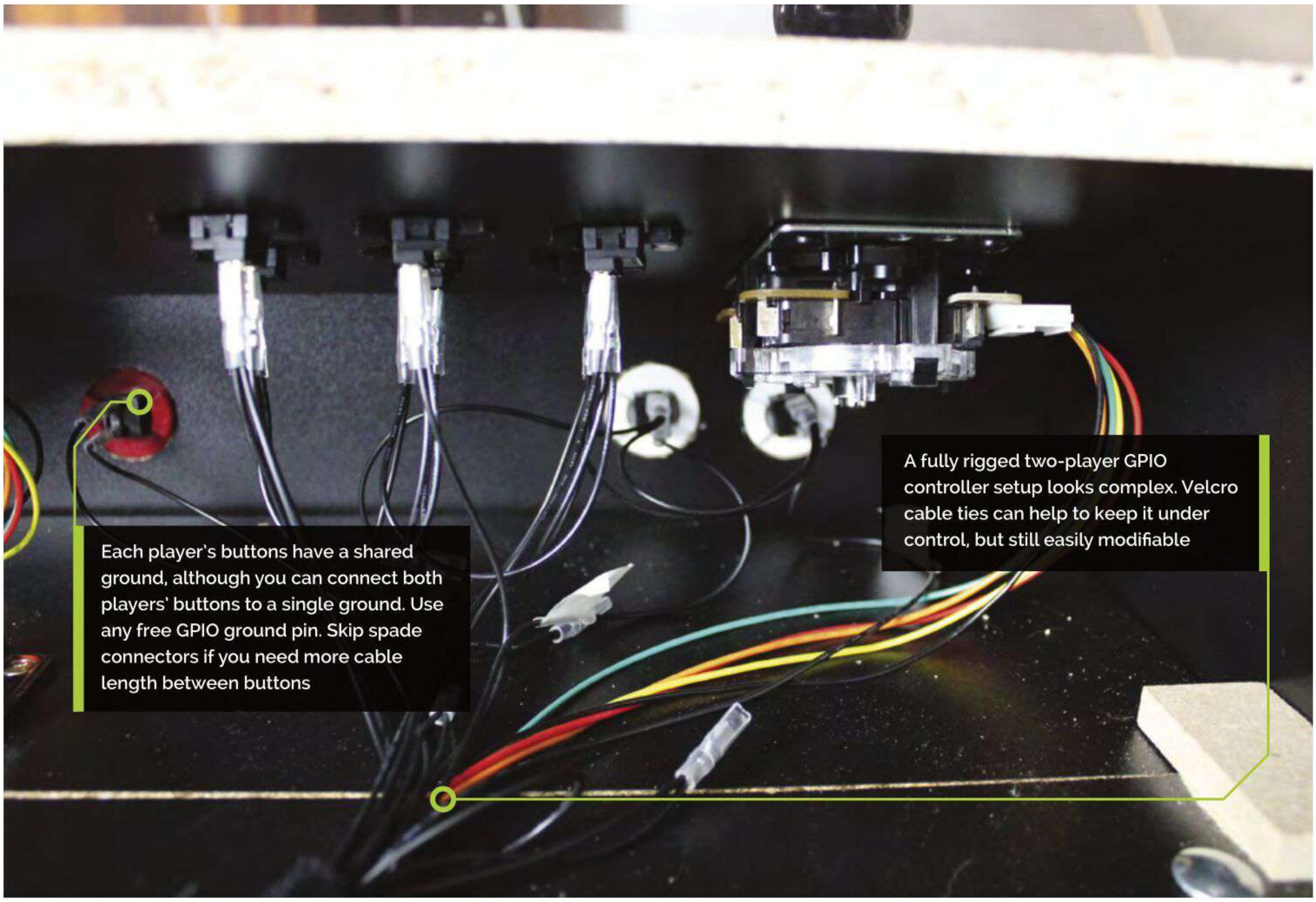
▼ Recalbox's main menu takes a couple of button presses to get to but has a comprehensive set of configuration options

02 Connect to GPIO

This is the fiddly bit. We suggest using a case for Raspberry Pi that fully exposes the GPIO pins. The GPIO wiring diagram shows which buttons, directional controls, and ground connections should be attached to each pin. While buttons and controls can be reconfigured in software, ground cannot. Our setup uses a total of 25 GPIO inputs, plus four ground connections. Input 25 is for a dedicated hotkey button.

03 Install and power up

Open Raspberry Pi Imager, connect your microSD card writer, and Choose OS > Emulation and game OS > Recalbox and the version of Recalbox that matches your Raspberry



Pi. Click Write and wait for the image file to be written to the microSD card. When Imager has finished, remove the microSD card and insert it into the Raspberry Pi in your arcade build. Connect the cabinet's monitor and speakers to Raspberry Pi. Plug in a keyboard on a long cable. Plug in Raspberry Pi's power and it will boot to Recalbox's EmulationStation interface, which you can immediately navigate using the keyboard. However, we still have to enable our GPIO arcade controls, wireless networking, and other configuration options.

04 Connecting Recalbox

Recalbox has SSH and Samba enabled by default, as well as a web interface available via your browser on **recalbox.local**. Recalbox should appear on your network as RECALBOX (File Sharing).

> Recalbox has SSH and Samba enabled by default, as well as a web interface

A wired Ethernet connection will give you immediate access to these. If you don't have one, press **ENTER** to open the menu, scroll to Main Menu, and select it with **A** on the keyboard, then select Network Settings, enable WiFi, select your SSID, and then select 'Wifi Key' to enter your password. Recalbox only has a root user. The default username is **root** and the password is **recalboxroot**.

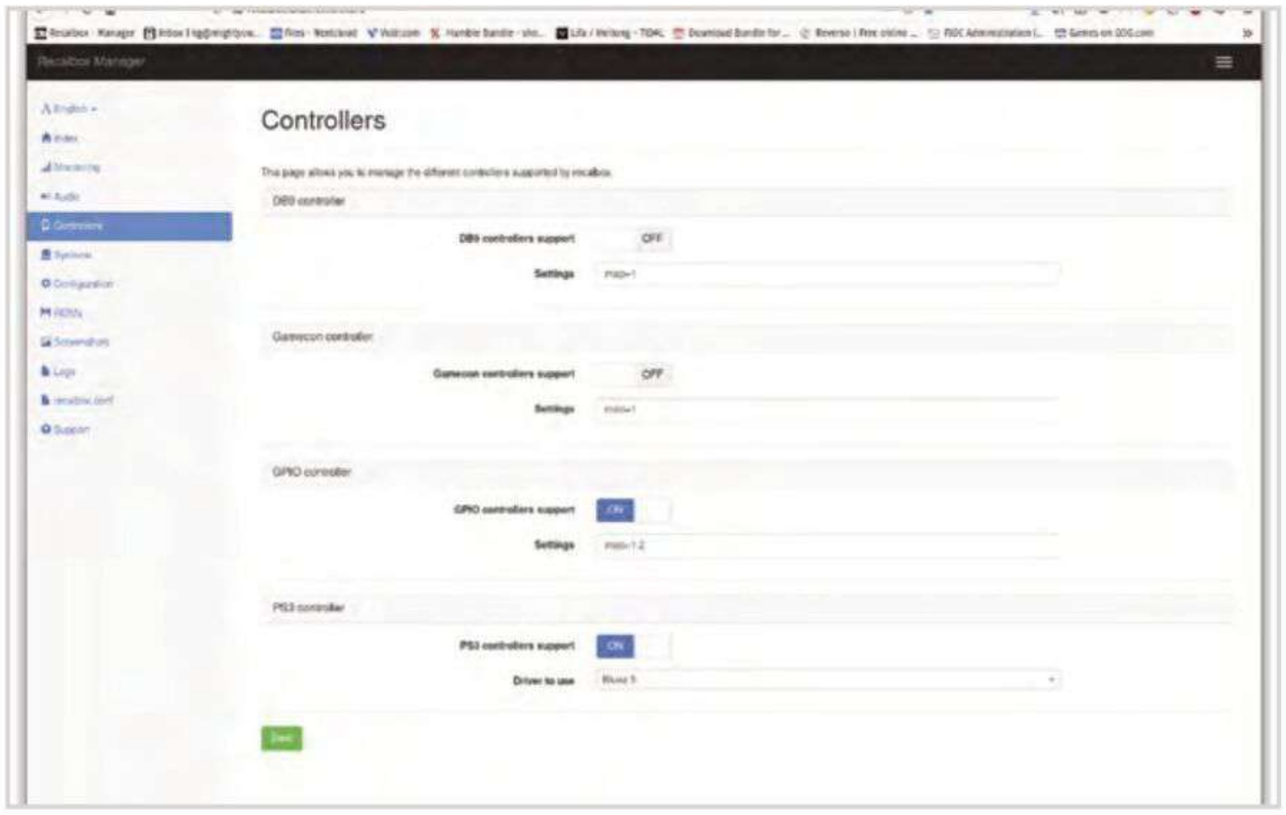
05 Configure Recalbox

You can access Recalbox's config file – **recalbox.conf** – by connecting via SSH, by browsing to the system directory in the Recalbox (File Sharing) Samba share, by pressing **F4** and then **ALT+F2** at the cabinet to exit to the console, or by going to **http://recalbox.local/** and selecting the **recalbox.conf** tab in the left-hand menu pane.

Under 'A - System options, Arcade metasystem', remove the semicolon that comments out `emulationstation.arcade=1`. This will make the arcade category the first entry in Recalbox's EmulationStation interface.

You'll Need

- Spade to DuPont cables
- Spade to DuPont shared ground cables
- Joystick to DuPont cables
- At least one Neo Geo Classics Collection game **magpi.cc/ironclad**



A web interface at http://recalbox.local gives you control over almost every aspect of your arcade machine's setup

Under D2 - GPIO controllers, set `controllers.gpio.enabled=1`. Save your changes and, at the arcade cabinet, open the menu, go to Quit > Fast restart Recalbox.

Top Tip

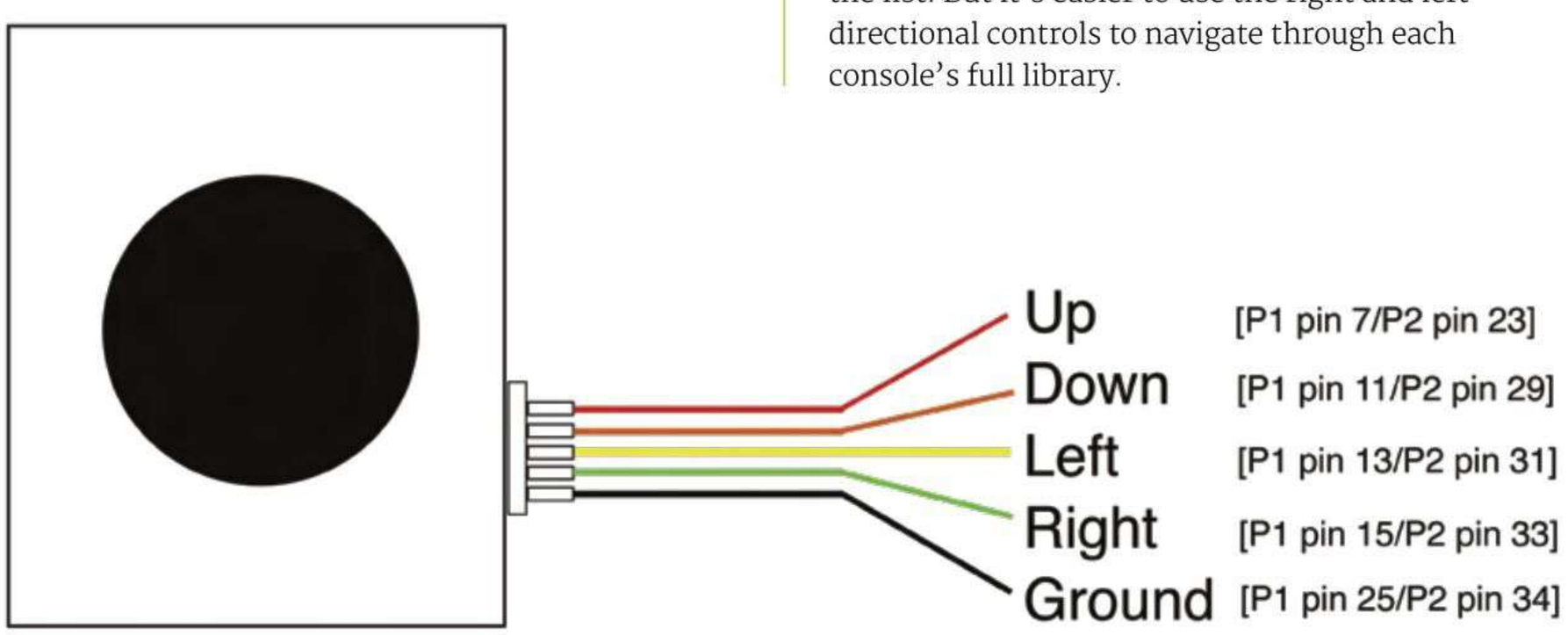
USB controls

To convert your controls to USB, use a Xinmotek board (**magpi.cc/xinmotec**) instead of connecting to GPIO.

06 Optional: Take control

Recalbox will now automatically detect GPIO controllers and, if all your buttons are wired as it expects, will already have the correct button configurations. Use the bottom-left button (B) to select options and the bottom-centre button (A) to go back. Left and right navigate between systems; up and down navigate between games or options within a menu. Press Start to open the configuration menu.

If your buttons aren't connected in that order, or if you prefer an alternative layout, open the menu and go to main menu > controllers settings > configure a controller. Press down to skip an entry that you don't have buttons for. If you don't have a hotkey button for one or more players, set it to Select.



Viewed from below, a standard Sanwa joystick's 5-pin connector goes to up, down, left, right, and ground. The diagram shows standard colour coding

07 Sounds good

If you have no sound, open the menu, select sound settings, and check the output device. We had to switch to 'headphones – analog' output to use our cabinet's speakers, connected to the 3.5 mm output on Raspberry Pi.

Recalbox plays background music all the time by default. This is charming, but a bit much for a cabinet that lives in the sitting room. Switch the Audio Mode to 'Video sound only' – to only hear the splash screens on boot – or 'No sound'.

If you prefer, you can add your own music by copying it to Recalbox's **share/music** directory.

08 Getting to know Recalbox

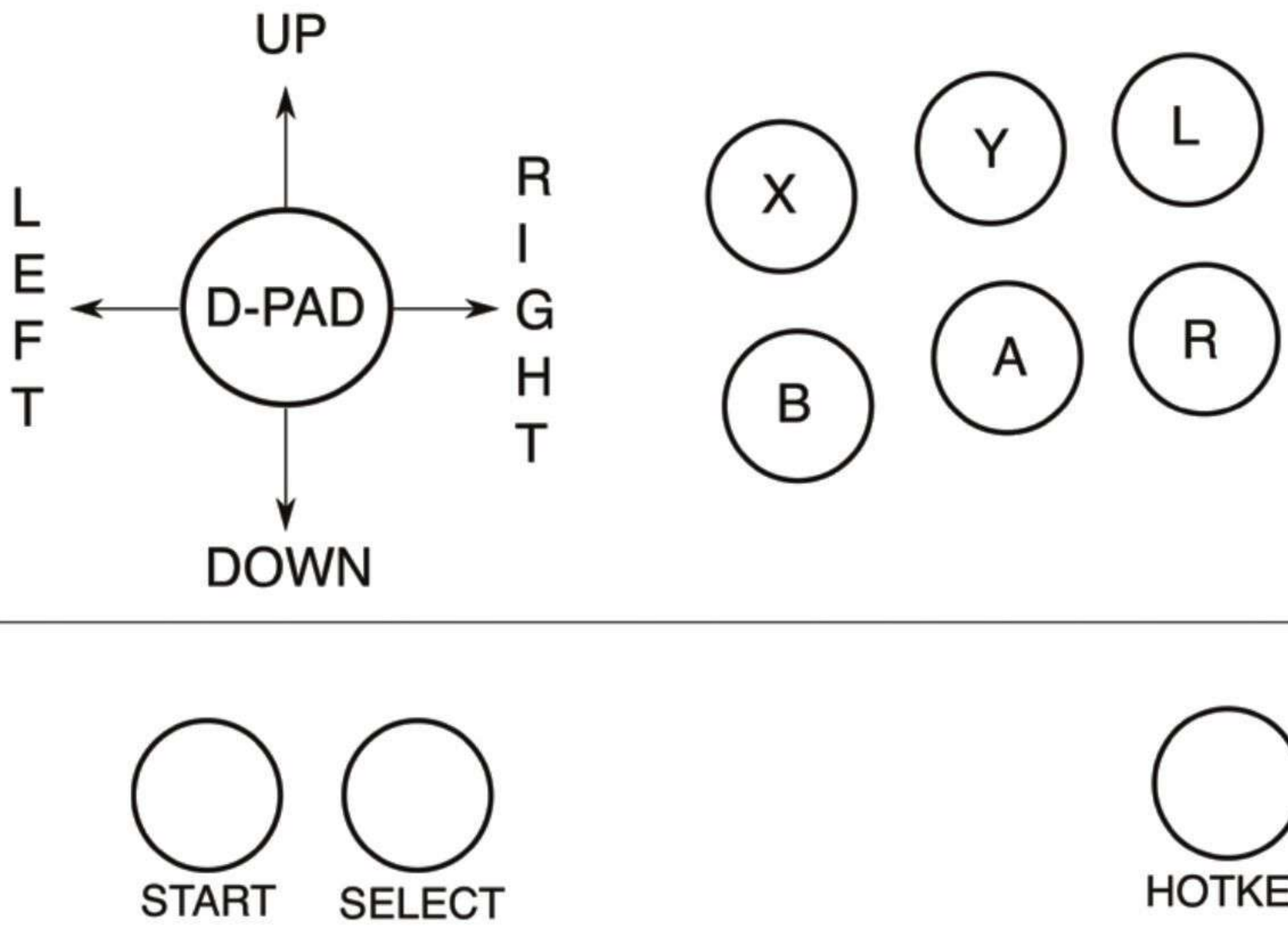
Recalbox comes preloaded with a number of freeware and open-source games. Because we enabled arcade mode, this category appears first. There are already four games loaded into it.

Select the category by pressing button B and scroll through them with the joystick. Gridlee, released in 1982, looks great for the era. Press B to load it.

Press Select to add credits and press Start when you're ready to play. When you've had enough, press the hotkey button and Start together to quit back to the Arcade menu.

> "Press the hotkey button and Start together to quit back to the Arcade menu"

You can press A to go back to the top menu, and use the joystick to navigate up and down through the list. But it's easier to use the right and left directional controls to navigate through each console's full library.



Button and joystick correspondences for player controls. The joystick maps to the D-pad. L and R correspond to L1 and R1, equivalent to the shoulder buttons of modern joypads

09 recalbox.local

Once your arcade machine is connected to your local network, you'll be able to access it in a web browser via **http://recalbox.local**. On the main page, you'll see shortcuts to a virtual gamepad, keyboard, and touchpad, which allow you to navigate through the arcade machine's menus remotely.

To add some authenticity to older titles, go to Systems and set the Shader set to Retro, which will apply community-favourite shader and scanlines settings to each game. On the other hand, if performance is poor, disable shaders and rewinding here. Click Save at the bottom of the page to store your changes.

Below, the Configuration tab lets you set networking options, enable and disable the Kodi media player, and configure the behaviour of the EmulationStation front end and the hotkey.

10 Manage game & BIOS files

The easiest way to manage your game ROMs on Recalbox is via the web interface, where the ROMs tab lets you select the directory for your desired console, stop the EmulationStation front end, upload games, and restart EmulationStation to load them.

You can also copy games over to the **roms** directory in Recalbox's Samba share. Even if you don't plan on emulating a specific console, don't delete the containing folders for its games, as they're required.

Recalbox also shares a **bios** directory, where you can add freeware or legally purchased computer or console BIOS files.

11 Buy and install a game

ROMs and a functional BIOS set for a number of Neo Geo maker SNK Corporation's games are available to buy as part of the Neo Geo Classics Collection (**magpi.cc/neogeoclassics**).

You'll need a Windows, macOS, or Linux PC to install or extract these. You'll find the ROM and BIOS files in the install directory; for example, **ironclad.zip** and **neogeo.zip** respectively for the fantastic scrolling shoot-'em-up Ironclad. If you don't want the whole collection, you can buy Ironclad alone at **magpi.cc/ironclad**.

Connect to Recalbox via SMB and copy the game ROMs into **roms**, and **neogeo.zip** into **bios**.

Restart EmulationStation and you should find your new games in the Arcade game list. Not all of them will work out of the box. Start any of them and press the hotkey and B buttons to open the Libretro emulation interface. Scroll down and select Options > Neo-Geo mode > Use UNIBIOS Bios. We aren't using UniBios here, but the file supplied by SNK is compatible with this setting.

Warning!
Mains electricity & power tools

Be careful when handling projects with mains electricity. Insulate your cables and disconnect power before touching them. Also, be careful when using power tools during this build.

magpi.cc/drillsafety
magpi.cc/electricalsafety

Top Tip

Preconfigured USB support

If your cabinet uses a USB controller board, then RetroPie won't need any extra drivers to detect your controls.

▲ SNK has made plenty of its arcade ROMs available to buy. Ironclad for Neo Geo-based arcade machines is a particular favourite

Press A twice to go back and select Resume. Your game should start.

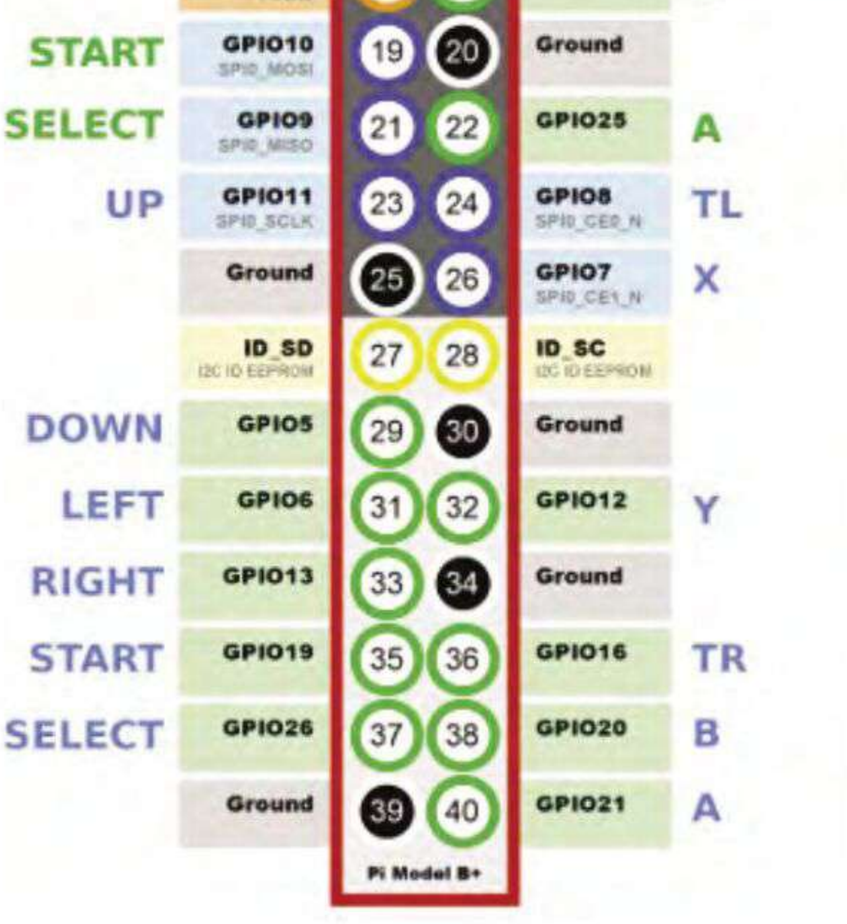
12 Tweak your games entries

To hide the games that come with Recalbox, from EmulationStation press Start > Main menu > Games settings > Hide preinstalled games. Unfortunately, you can't pick and choose which get hidden, but you can manually download and re-add any that you'd like to keep.

You can also disable the ports category by editing **recalbox.conf** to include:

```
emulationstation.collection.ports=1
emulationstation.collection.ports.hide=1
```

If you want to add images or change the titles of the games you've added to Recalbox, the easiest approach is to use the built-in scraper. Highlight the game in the menu, press Start > Edit game > Scrape. You can also add your own ratings and keywords in this menu.



▶ GPIO wiring: Connect your joysticks and buttons to Raspberry Pi's GPIO as shown. *Image by digitalLumberjack of the Recalbox project, licensed under GPL2*

Warning!
Copyright alert!

It is illegal to download copyrighted game or BIOS ROMs in the UK without the permission of the copyright holder. Only use official purchased or freeware ROMs that are offered for download with the consent of the rights holder.

magpi.cc/legalroms

13 Get more games

The creators of the MAME emulator have been given permission to distribute some early arcade games, which you can find for download at **magpi.cc/mameroms**.

Many other emulated arcade games have been released for use on modern computers, but some – including collections by SNK, Capcom, Irem, and Namco – require an additional extraction and re-bundling stage. You can find tools and game lists to help you buy and use these at RED-project (**magpi.cc/redproject**) and SF30ac-extractor (**magpi.cc/sf30ac**). Linux GOG users may also require innoextract (**magpi.cc/innoextract**). Non-Neo Geo arcade games should go into the **roms/MAME** directory.

The homebrew scenes for arcade games tend to focus on physical releases, but we've had luck with Codename: Blut Engel for Neo Geo and Santaball (**magpi.cc/neogeohomebrew**) for Neo Geo CD.

For more retro and homebrew games that work well with arcade controls, including Sega's Mega Drive Classics collection, see **magpi.cc/legalgameemu** and **magpi.cc/legalroms**.

Wireframe

Join us as we lift the lid on video games

Visit **wfmag.cc** to learn more

Part 02

Configure Pi-hole's DNS settings

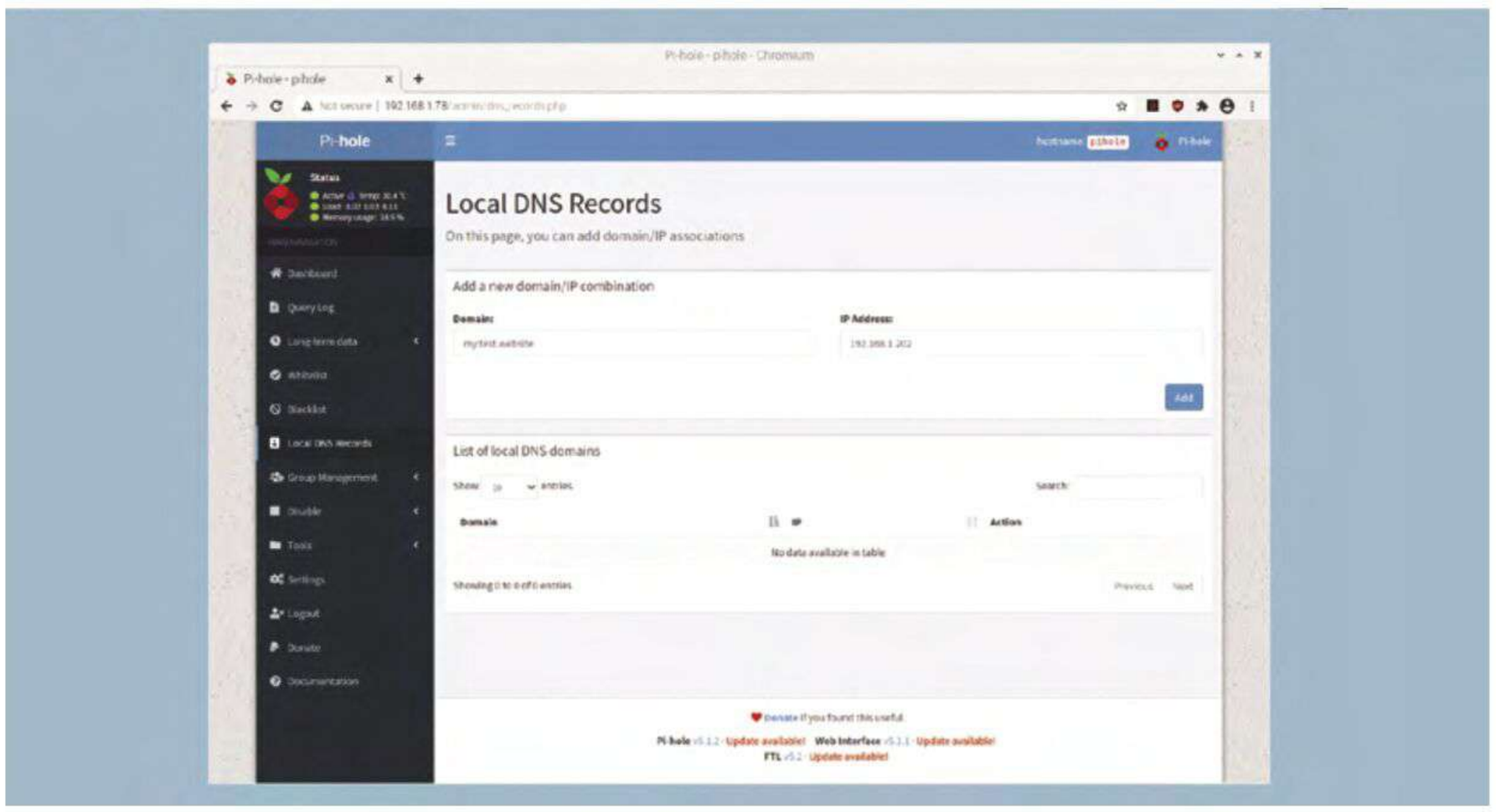
Beef up your privacy and give computers on your network their own web addresses by tweaking Pi-hole's DNS settings

MAKER

Nik Rawlinson

Esperanto-speaking, pencil-wielding, single-board computing fan who likes hyphens and remembers what that icon on the save button depicts.

nikrawlinson.com



If you're hosting or developing a website on your local network, Pi-hole lets you assign a memorable name in place of its IP address

In the previous tutorial, we set up Pi-hole to filter web content. The result was less cluttered web pages, free of ads, that loaded more quickly. Here, we'll take things further by setting up a local DNS (Domain Name System) server so Pi-hole can bypass third-party DNS providers to find a direct route to the web resources we need, for greater security. We'll also set up some friendly web addresses for computers on our local network. That way, if we're using them for tasks like building and testing websites, we can access them directly through a browser without having to use their numeric IP addresses.

You'll Need

- Pi-hole **pi-hole.net**
- Web browser
- Optionally: Terminal access

01 Log in to Pi-hole

Open a new browser window and log in to your Raspberry Pi that's hosting Pi-hole. If you're using a computer that's configured to use Pi-hole to filter its content, you'll find it by entering:

http://pi.hole/admin

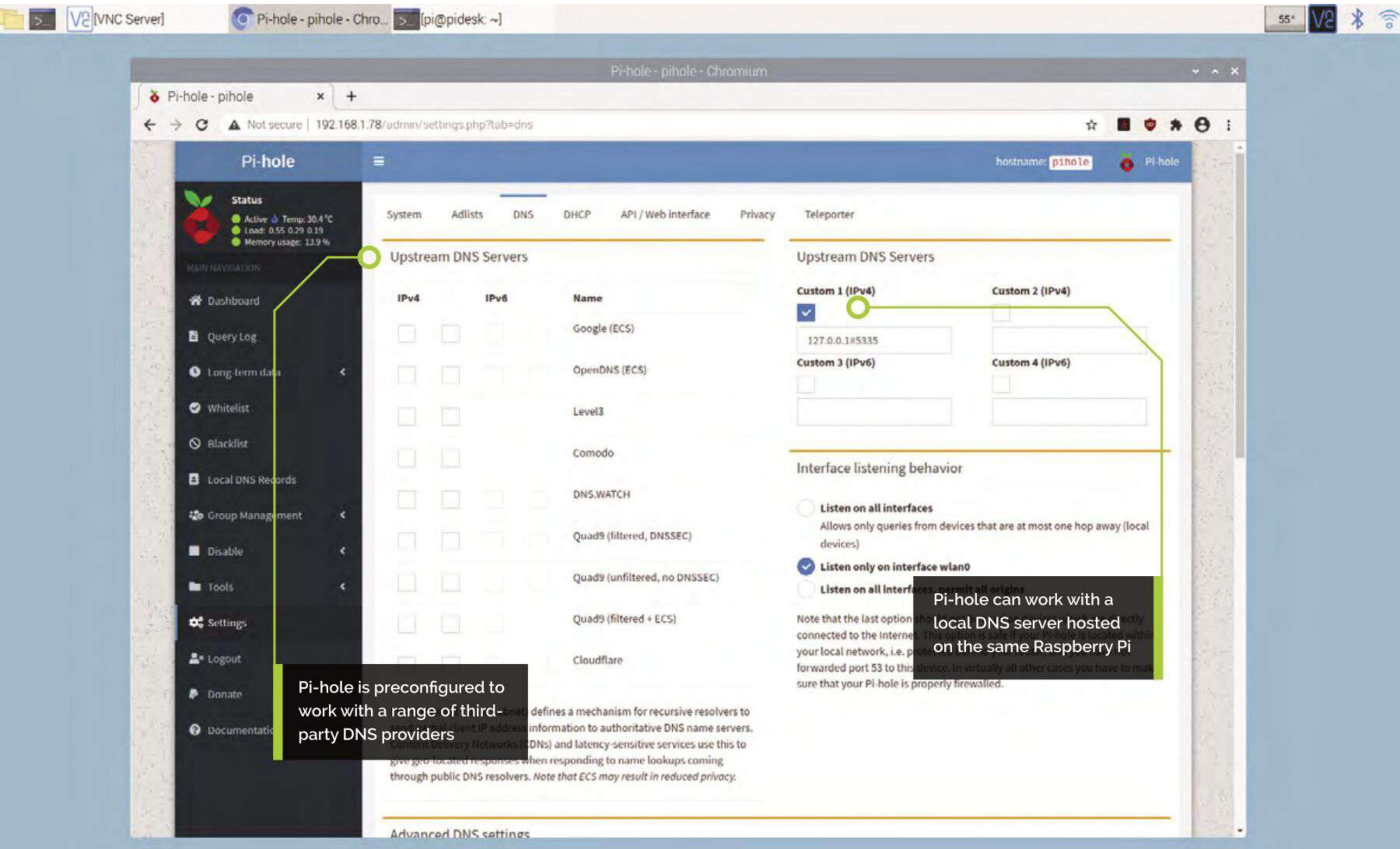
If not, you'll need to use the Pi-hole Raspberry Pi's IP address, which you set up during installation. For example:

http://192.168.1.148/admin

There's no username, but you will need your Pi-hole password. Click Login and enter it.

02 Creating local DNS records

You'll notice that one of the options for logging into Pi-hole was to type **pi.hole** into the



browser, rather than use the Raspberry Pi's IP address. This works because when your computer passes the request to Pi-hole, it first looks for a matching local record before passing it on to an external DNS server. Finding a match, it checks the address of the host that it points to and delivers whatever result it finds there. You can apply the same trick to give web addresses to any computer on your network.

> Choose a memorable address and type it into the Domain box

03 Friendly names, locally

If you're running a web server on one of your computers, make a note of its IP address. Now click Local DNS Records in the Pi-hole sidebar. Type the numeric address of the computer hosting your development server in the IP Address box in the 'Add a new domain/IP combination' panel. Choose a memorable address and type it into the Domain box; for example, 'my.test.site'. Click Add and, within a couple of seconds, the DNS table will be updated. Open a browser window on any device filtered by Pi-hole and type the new address, prefixed by **http://**.

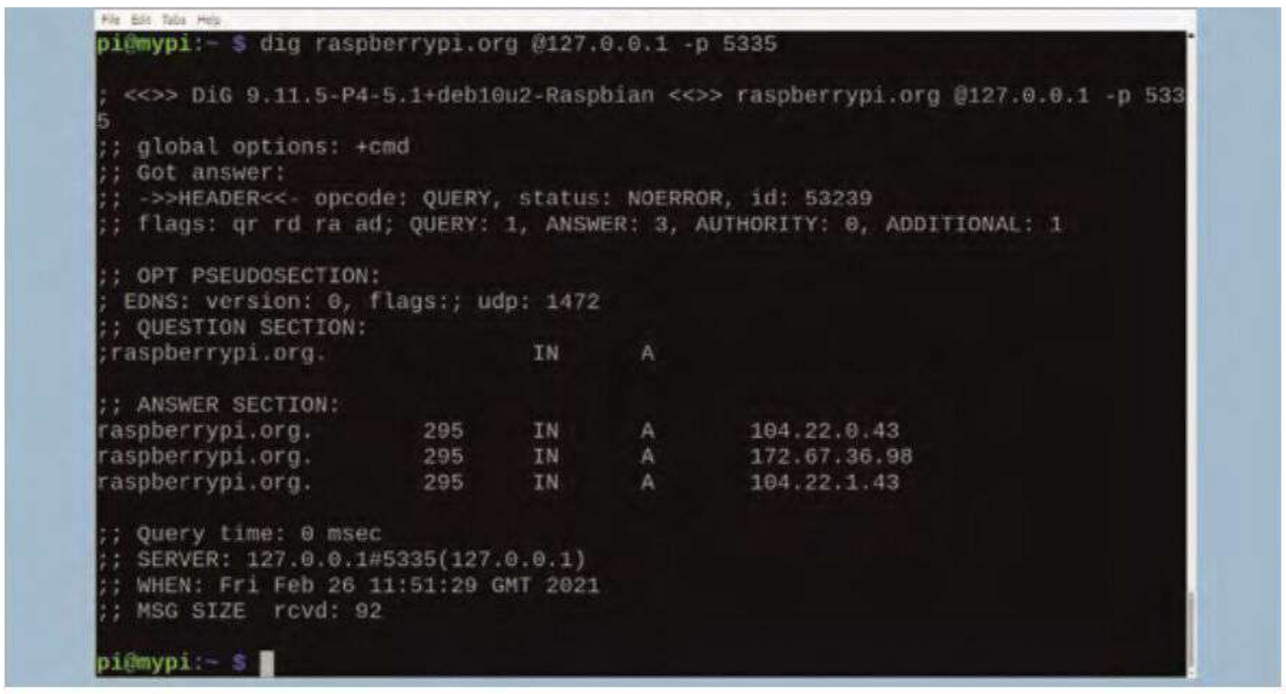
04 Switch DNS server

When setting up Pi-hole, you selected an external DNS provider, which is used to look up the addresses of permitted web-hosted assets. If you want to change this, click Settings in the Pi-hole sidebar, followed by the DNS tab. You'll notice that at least two ticks appear alongside your preferred service (four if you're filtering both IPv4 and IPv6 requests), for the primary and secondary servers. If you choose to use a secondary provider, either swap both ticks, as appropriate, or mix and match. You could, for example, use Google and Cloudflare.

Top Tip

Get started

The first part of this tutorial series appeared in *The MagPi* magazine issue 104 **magpi.cc/104**



▲ When you've set up Unbound, use the dig command in the Terminal to check that it can locate online resources

If you choose an alternative provider, it will use whichever DNS responds first (see **magpi.cc/roundrobindns** for more information).

Top Tip

Reliable connections

Consider using Ethernet, rather than WiFi, if your Raspberry Pi has it, to avoid DNS errors on your network caused by weak wireless signals.

05 Horses for courses

Different DNS services support different features. Cloudflare, for example, makes a point of not logging the IP addresses of browsers using its DNS server, and both OpenDNS and Comodo maintain lists of phishing sites, which should be blocked in addition to anything on your Pi-hole blacklist. OpenDNS Family Shield, which doesn't appear in the list of presets, not only blocks malware, but pornography too. To enable it, tick the checkboxes beneath 'Custom 1 (IPv4)' and 'Custom 2 (IPv4)' in the right-hand box and enter the addresses **208.67.222.222** and **208.67.220.220** in the input boxes. Scroll down and click Save.

▼ Pi-hole lets you choose your preferred DNS provider during setup, but you can switch to an alternative through the Dashboard

06 A local DNS server

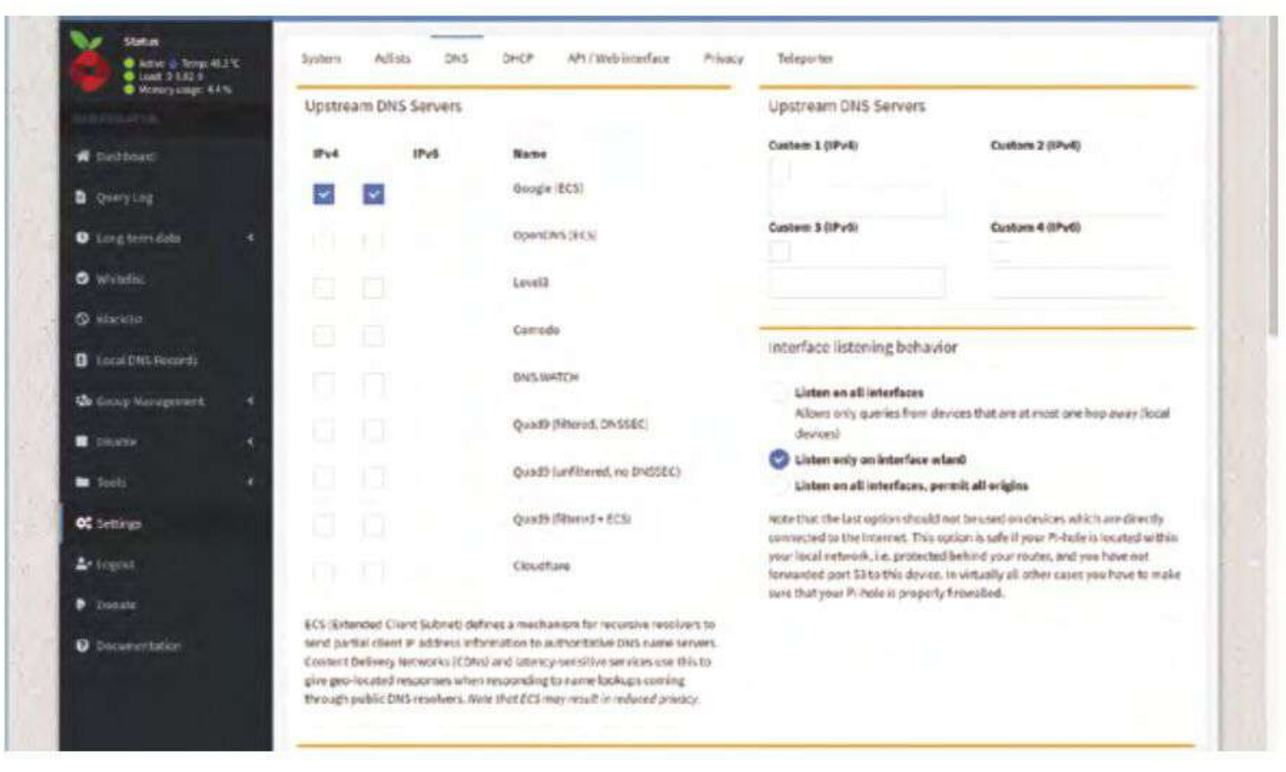
Alternatively, you may choose to run your own local DNS server. Why? Because more security-conscious system admins can be wary of trusting third-party providers rather than referring back to the most definitive sources for every address the computers on their network request. Pi-hole makes this possible by piggybacking the open-source Unbound server app. This isn't installed by default as part of Raspberry Pi OS, so open a Terminal window on the device hosting Pi-hole and type `sudo apt install unbound`. When asked to confirm the installation, press **Y**.

07 Configure Unbound

When the installation has completed, type `sudo nano /etc/unbound/unbound.conf.d/pi-hole.conf` and press **ENTER**. Pi-hole's developers have helpfully provided a complete configuration file at **magpi.cc/piholeunbound**. Copy everything in the large grey box in the Configure Unbound section, then press **CTRL+X** to quit Nano. Confirm that you want to save the configuration file when asked, and accept the suggested file name. Now restart the DNS server by typing `sudo service unbound restart` and pressing **ENTER**. Your Raspberry Pi is now running its own DNS server that interrogates the internet's root servers, rather than preconfigured DNS servers, to locate the web resources you need.

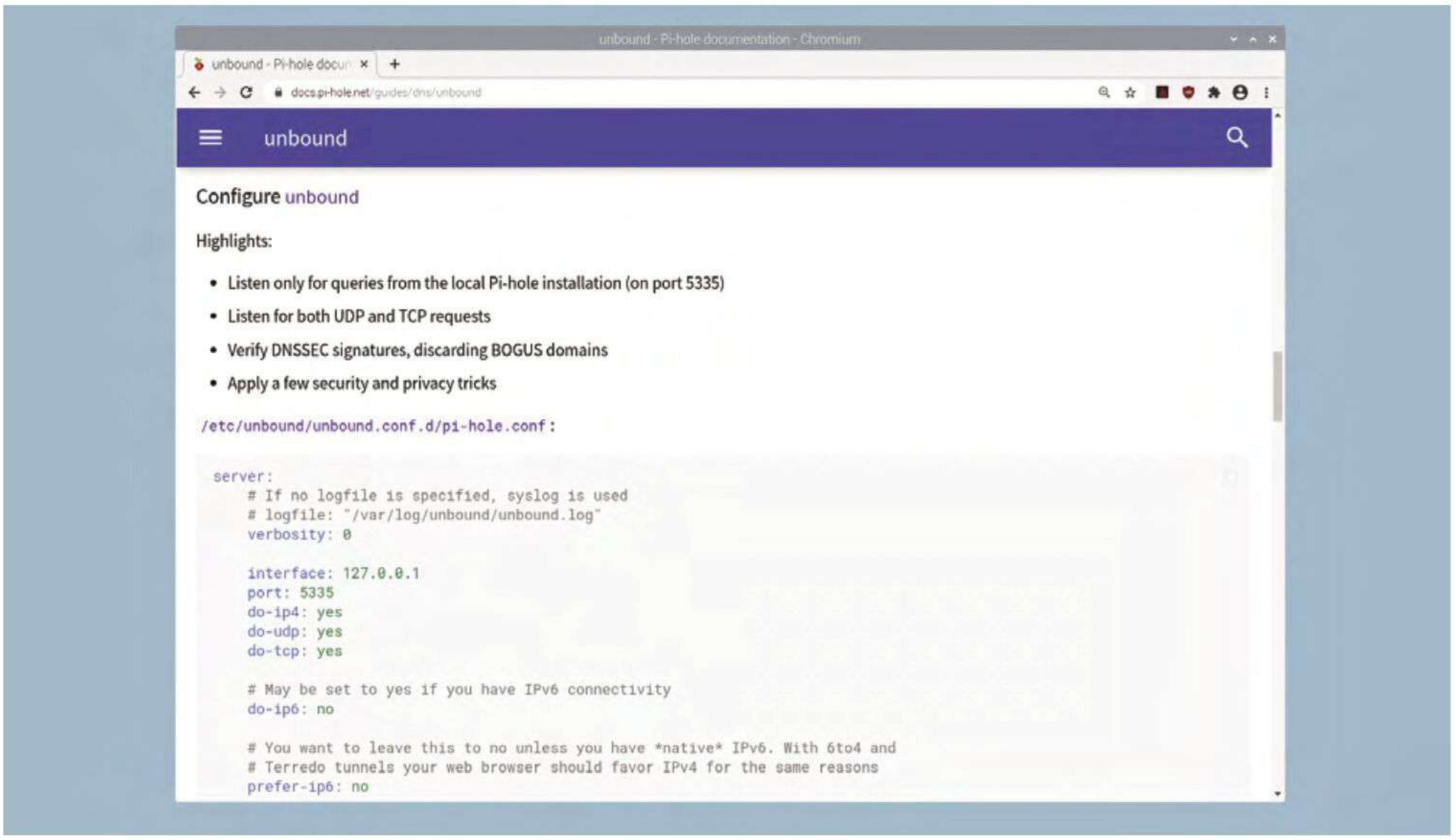
08 Test your DNS server

Test Unbound by typing `dig raspberrypi.org @127.0.0.1 -p 5335`. This invokes the Linux dig tool to query the DNS server hosted on port 5335 at the IP address 127.0.0.1 for the record that relates to **raspberrypi.org**. The address 127.0.0.1 effectively loops whatever is pointed at it back on itself, whether it's a browser request or, in this case, a command. Within a second, Unbound will have found the servers hosting the Raspberry Pi website and returned their IP addresses.



09 Check DNS caching

Once Unbound has identified the numeric addresses that relate to the web address you fed it, it saves them to its local database so that the next time the same resource is requested it doesn't need to repeat the operation. Press the up arrow key to reload the command you just typed and press



▲ Pi-hole's help pages include a comprehensive configuration file for using Unbound, which you can paste directly into your own config files

ENTER, and you may notice that the response is slightly faster the second time around. That's because Unbound is what's known as a caching server: it caches its work for future reuse. You now need to tell Pi-hole to use this instead of one of its preset DNS options.

10 Enable your DNS server

Return to the Pi-hole dashboard's DNS settings. Uncheck your existing DNS servers in the left panel, and check the box beside 'Custom 1 (IPv4)' in the right panel. In the box below, type '127.0.0.1#5335', then scroll to the bottom of the page and click Save. Make sure you use a hash (#), rather than the more usual colon (:) between the IP address and port number. When the configuration page refreshes, open a browser window on any computer filtered by Pi-hole and visit a website to check that everything is working properly.

11 Flush your DNS data

If you've made changes to your DNS settings – and particularly if you've set Pi-hole to redirect bespoke addresses to computers on your network, or to use an alternative DNS service with child-friendly filtering – but you're still seeing content on connected computers that should be blocked, flush the DNS cache on your computer. On a Windows machine, open a Command Prompt window and type `ipconfig /flushdns`. On a Mac, open a Terminal window and type `dscacheutil -flushcache` and press **ENTER**, then type `sudo killall -HUP mDNSResponder` and press **ENTER**. On a Mac, you'll need to provide your password.

12 Back up Pi-hole

Now that you've made changes to your Pi-hole configuration, you should make a backup of your settings so you can reinstate them should anything corrupt the system. Click Settings in the Dashboard's sidebar, followed by the Teleported tab. Click the Backup button and save the resulting file somewhere safe. To later reinstate it, click the 'Choose file' button in the Restore panel, select the downloaded file, and click Restore. Make a backup every month or so to ensure your most recent settings, blacklists, and whitelists are reflected in the archive.

Top Tip

Minimal install

The Dashboard is available across your network, which means you can run Pi-hole on Raspberry Pi OS Lite.

Part 06

Create GUIs with Python: Destroy the dots

Learn how to use a Waffle to create a tasty game

MAKER

Laura Sach

Laura leads the A Level team at the Raspberry Pi Foundation, creating resources for students to learn about Computer Science.

@CodeBoom

MAKER

Martin O'Hanlon

Martin works in the learning team at the Raspberry Pi Foundation, where he creates online courses, projects, and learning resources.

@martinohanlon

You saw in the Tic-tac-toe game how to create a GUI on a grid layout in order to present the player with a grid-like board (magpi.cc/105). If you're making a game involving a larger grid, there is a type of guizero widget called a Waffle which can instantly create a grid for you, and is really useful for creating all kinds of games.

A Waffle was originally a grid of squares in early versions of guizero. This game is called 'Destroy the dots' and it came about because Martin thought it was a good idea to allow a Waffle widget to contain a mixture of squares and dots.

Aim of the game

In this game, you need to destroy the dots before they destroy you! The board consists of a grid of squares. The squares will gradually turn into dots. To destroy a dot, click on the dot and it will turn back into a square. The aim of the game is to last as long as possible before being overrun by dots (**Figure 1**).

Set up the game

Let's start by making a guizero program which contains the instructions for the game and a Waffle. By now you should be familiar with the layout of a standard guizero program with sections for imports, functions, variables, and the app itself.

First, create an App and inside it add a Text widget for the instructions and a Waffle widget for the board:

Figure 1

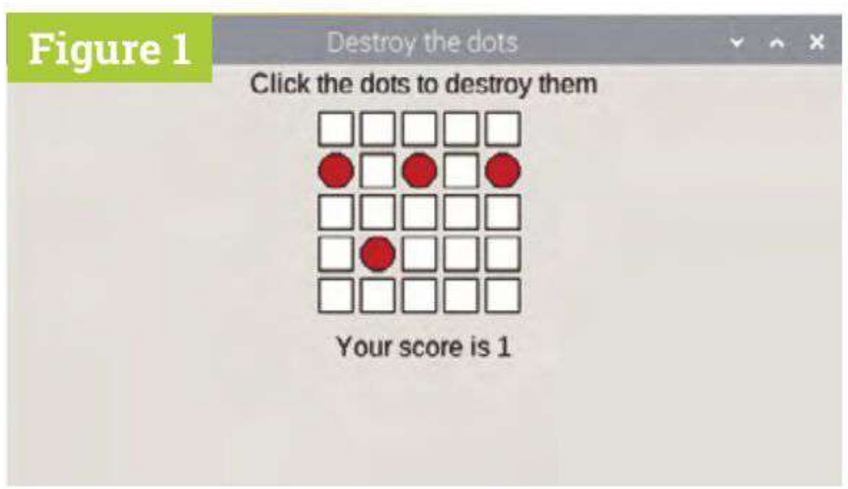


Figure 1 Destroy the red dots before they take over the board

```
# Imports --------------
from guizero import App, Text, Waffle

# App ------------------
app = App("Destroy the dots")

instructions = Text(app, text="Click the
dots to destroy them")
board = Waffle(app)

app.display()
```

If you run your program, you will see a small 3×3 grid of white squares. If you want to make your grid bigger than this, you can add width and height properties to your Waffle:

```
board = Waffle(app, width=5, height=5)
```

Your code should now resemble that in the **destroy1.py** listing.

Bring on the dots

Next you need to write a function to find a random square on the board and turn it into a dot. Begin a new function in your functions section called `add_dot()`:

```
def add_dot():
```

To choose a random square on the board, you need to be able to generate a random pair of integers as co-ordinates. Add a line in your imports section to import the `randint` function from the `random` library, which lets you generate a random integer.

```
from random import randint
```

Let's generate two variables, x and y, which you can use to reference a co-ordinate on the grid. Inside your `add_dot()` function, begin your code like this:

```
x, y = randint(0,4), randint(0,4)
```

Notice that you have generated two random integers between 0 and 4, because earlier on you set the width and height of the grid to be 5 – the rows and columns will be numbered from 0. If you chose different values earlier on, you will need to adjust the values here to fit the size of your grid. However, there is a better way to manage aspects like this (see 'Using constants' box on page 55).

Dot or not?

Now that you know about constants, you can use the following function to generate a random co-ordinate on the grid:

```
def add_dot():
    x, y = randint(0,GRID_SIZE-1),
randint(0,GRID_SIZE-1)
```

At this point, you don't know whether the randomly chosen co-ordinate is already a dot or not. This might not seem to make any difference at the start of a game when the board is mostly squares, but as the board gets filled up with dots, you need to make sure that the space is actually a square, otherwise the game will be too easy. One way to achieve this is to run a loop which checks whether the chosen square is already a dot, and if it is, chooses another random square:

```
x, y = randint(0,GRID_SIZE-1),
randint(0,GRID_SIZE-1)
while board[x, y].dotty == True:
    x, y = randint(0,GRID_SIZE-1),
randint(0,GRID_SIZE-1)
```

You might realise that this isn't a particularly efficient method of choosing a random square that is not a dot, but it will do for what we need in this game. As soon as this loop finishes, you can be sure that the randomly chosen x, y co-ordinate is definitely a square. Let's convert it to a red dot – following (not inside) your `while` loop, add the following lines:

```
board[x, y].dotty = True
board.set_pixel(x, y, "red")
```

Add a call to your new `add_dot()` function in the app section after you've created the board. Your program should now resemble **destroy2.py**. When you run it, you should see a single random red dot in the grid. If you run the program again,

destroy1.py

DOWNLOAD THE FULL CODE: magpi.cc/guizerocode

Language: **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, Text, Waffle
004.
005.  # Variables -------------
006.
007.  # Functions -------------
008.
009.  # App -------------------
010.
011.  app = App("Destroy the dots")
012.
013.  instructions = Text(app, text="Click the dots to destroy them")
014.  board = Waffle(app, width=5, height=5)
015.
016.  app.display()
```

> At this point, you don't know whether the randomly chosen co-ordinate is already a dot or not

destroy2.py

Language: **Python 3**

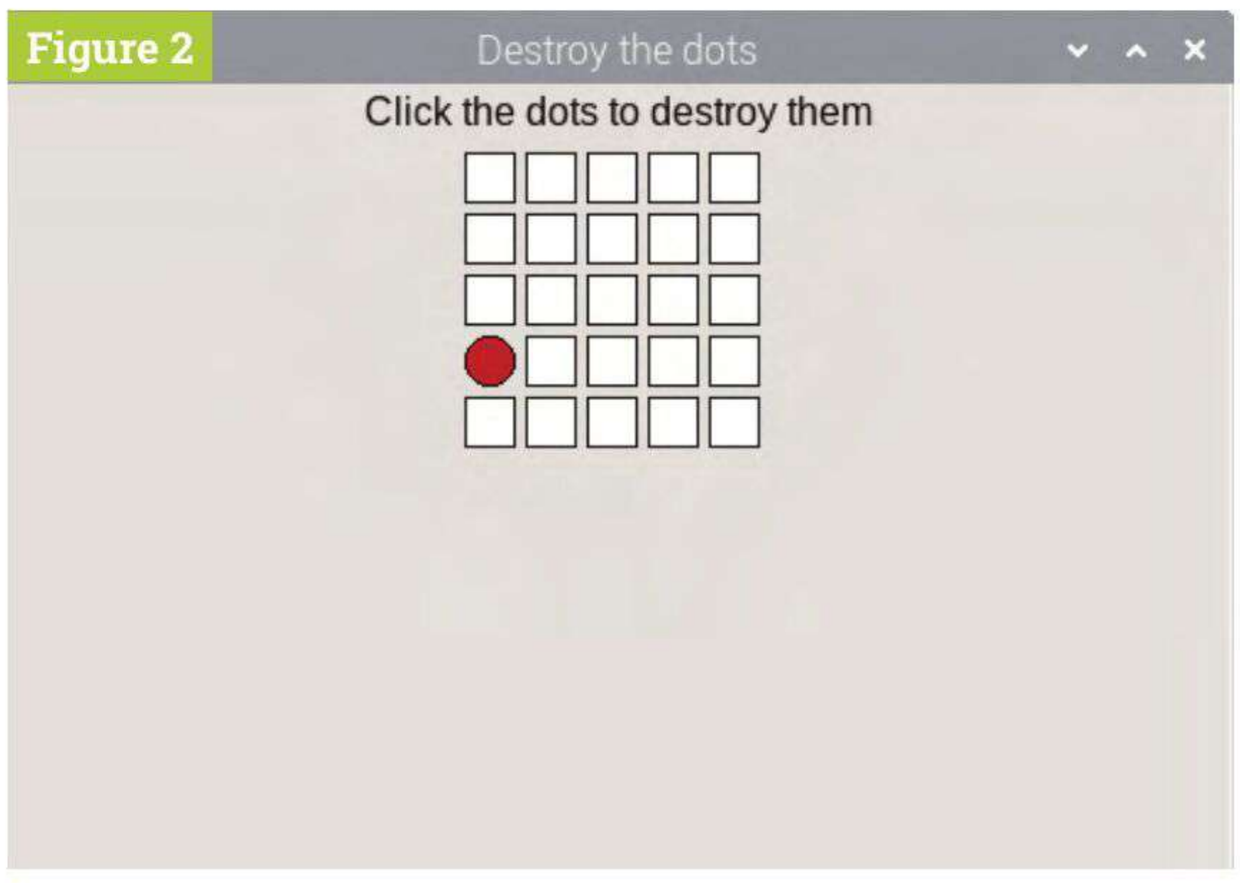
```
001.  # Imports ---------------
002.
003.  from guizero import App, Text, Waffle
004.  from random import randint
005.
006.  # Variables -------------
007.
008.  GRID_SIZE = 5
009.
010.  # Functions -------------
011.
012.  def add_dot():
013.      x, y = randint(0,GRID_SIZE-1), randint(0,GRID_SIZE-1)
014.      while board[x, y].dotty == True:
015.          x, y = randint(0,GRID_SIZE-1), randint(0,GRID_SIZE-1)
016.      board[x, y].dotty = True
017.      board.set_pixel(x, y, "red")
018.
019.  # App -------------------
020.
021.  app = App("Destroy the dots")
022.
023.  instructions = Text(app, text="Click the dots to destroy them")
024.  board = Waffle(app, width=5, height=5)
025.  add_dot()
026.
027.  app.display()
```

destroy3.py

Language: **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, Text, Waffle
004.  from random import randint
005.
006.  # Variables -------------
007.
008.  GRID_SIZE = 5
009.
010.  # Functions -------------
011.
012.  def add_dot():
013.      x, y = randint(0,GRID_SIZE-1), randint(0,GRID_SIZE-1)
014.      while board[x, y].dotty == True:
015.          x, y = randint(0,GRID_SIZE-1), randint(0,GRID_SIZE-1)
016.      board[x, y].dotty = True
017.      board.set_pixel(x, y, "red")
018.
019.  def destroy_dot(x, y):
020.      if board[x,y].dotty == True:
021.          board[x,y].dotty = False
022.          board.set_pixel(x, y, "white")
023.
024.
025.  # App -------------------
026.
027.  app = App("Destroy the dots")
028.
029.  instructions = Text(app, text="Click the dots to destroy them")
030.  board = Waffle(app, width=GRID_SIZE, height=GRID_SIZE,
031.  command=destroy_dot)
032.  add_dot()
033.
034.  app.display()
```

Figure 2



▲ **Figure 2** Generating a random red dot

the dot will probably be in a different random place (**Figure 2**).

Destroy the dot

So far there is only one dot – let's destroy it! Don't worry: you'll add more dots to destroy later on, but once you can destroy one, you can destroy them all!

Make a new function in your functions section with a really satisfying name – `destroy_dot` – and give it two parameters, x and y.

```
def destroy_dot(x, y):
```

This function will check whether the co-ordinate x,y is a dot (rather than a square). You can do this using the same code as the code to create a dot – the code `board[x, y].dotty` will return True if that coordinate is a dot, and False if it is a square.

```
if board[x,y].dotty == True:
```

If the co-ordinate is a dot, change it to a square by setting its `dotty` property to False, and also change its colour back to white:

```
if board[x,y].dotty == True:
    board[x,y].dotty = False
    board.set_pixel(x, y, "white")
```

This function needs to be triggered whenever the board is clicked. Find the line of code you already have which creates the board, and add a command like this:

```
board = Waffle(app, width=GRID_SIZE,
height=GRID_SIZE, command=destroy_dot)
```

This will call the `destroy_dot` function whenever a space on the board is clicked.

Note that a Waffle widget will automatically pass two parameters to any command function; these will always be the x and y co-ordinates of the pixel that was clicked on to trigger the command.

Your code should now look like **destroy3.py**. Test your program by running it and clicking on the dot. You should see the dot turn back into a white square. If you click on a square that is not a dot, nothing should happen.

More dots!

Now it's time to actually make the game a challenge, by adding continually spawning dots. Let's start off by adding a new random dot every second. To do this, you need to schedule a call to

the `add_dot` function every second using a built-in feature of guizero called `after`.

In your app section, remove the call to `add_dot()` and replace it with a new line of code:

```
board.after(1000, add_dot)
```

This line of code means 'after 1000 milliseconds (1 second), call the function `add_dot`'.

If you run the program now, you'll still get a single dot, but it will appear on the grid after a delay of 1 second. Here's the clever bit. Find your `add_dot` function and add the same line of code to it, at the end of the function.

This will schedule a new call to `add_dot` every time a new dot finishes being added. The next dot is scheduled to appear in 1 second as well, so if you run the program you should see a new dot appearing on the grid every second (**Figure 3**).

Try running your program, which should now look like **destroy4.py**. Since you already wrote the method to destroy a dot, clicking on any dot should remove it. However, if you play the game for a while you will notice it is pretty easy to keep up with the pace of one dot every second and it is almost impossible to lose the game.

You still need to add two things – a score to keep track of how many dots you have destroyed, and a way of making the game get more difficult so that it becomes a challenge.

Add a score

Adding a score is pretty straightforward and takes three steps:

- Add a variable to keep track of the score; the variable should start at 0.
- Display a message on the GUI with the current score.
- Any time the `destroy_dot` function is called and a dot is destroyed, add 1 to score and update the message display.

Try to add the code yourself using what you have already learnt.

Hint: *To update the score variable from the* `destroy_dot` *function, you will need to declare it a global.*

Hint: *If you get an error saying that the variable score is referenced before assignment, make sure your variables section comes before your functions section.*

The solution is shown overleaf if you are stuck...

Using constants

Setting the height and width of your Waffle to 5 is known as using a 'magic number' in a program, because the specific number is hard-coded into the program. If you want to change the size of the grid, you will need to find everywhere in the program this number appears and change it, which might be messy.

Better programming practice would be to define a constant in your variables section called **GRID_SIZE** and set it equal to 5:

```
GRID_SIZE = 5
```

Then, instead of defining your Waffle's dimensions with a magic number 5, you can put:

```
board = Waffle(app, width=GRID_SIZE, height=GRID_SIZE)
```

If you subsequently decide to change the size of the grid, you can just change the value of this constant. Thinking about this type of thing at the time you write the program will help you to avoid headaches later if you decide to change it.

destroy4.py

Language: **Python 3**

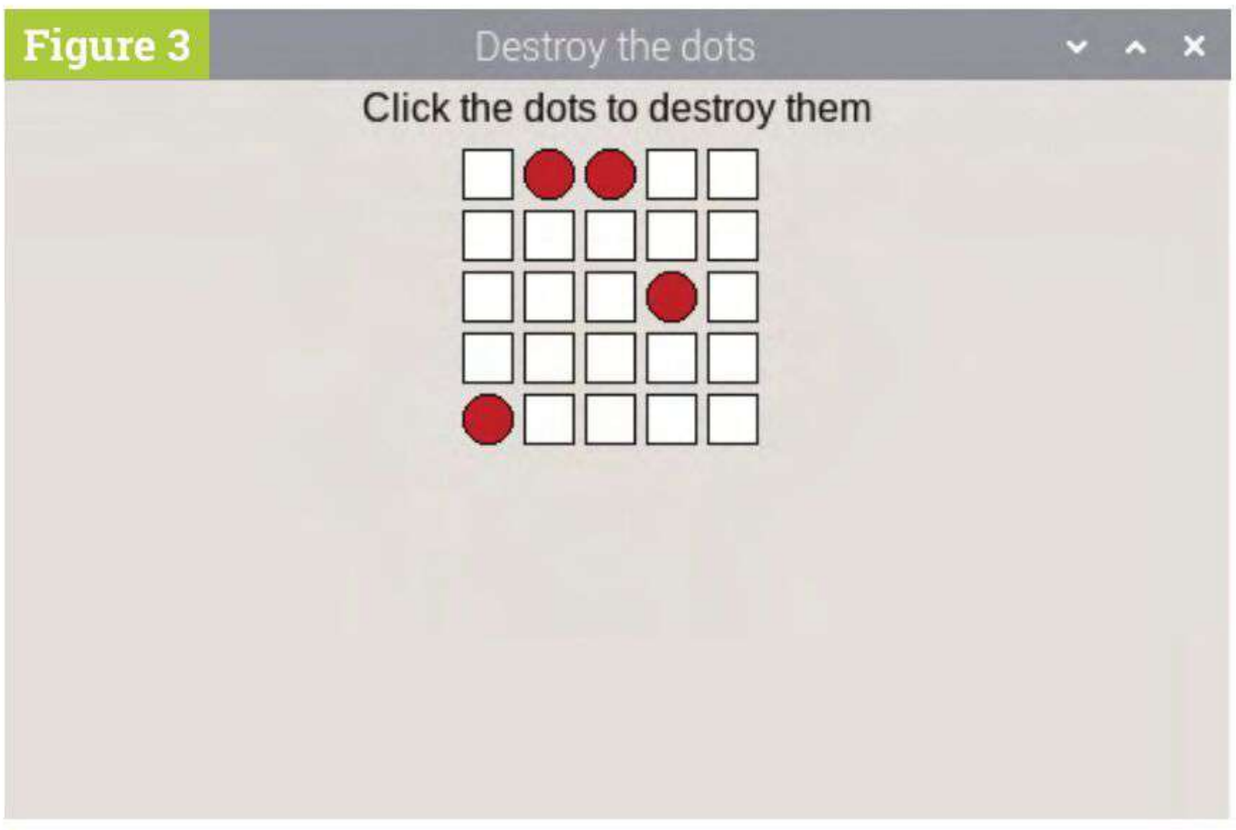
```
# Imports --------------

from guizero import App, Text, Waffle
from random import randint

# Variables ------------

GRID_SIZE = 5

# Functions ------------

def add_dot():
    x, y = randint(0,GRID_SIZE-1), randint(0,GRID_SIZE-1)
    while board[x, y].dotty == True:
        x, y = randint(0,GRID_SIZE-1), randint(0,GRID_SIZE-1)
    board[x, y].dotty = True
    board.set_pixel(x, y, "red")
    board.after(1000, add_dot)

def destroy_dot(x,y):
    if board[x,y].dotty == True:
        board[x,y].dotty = False
        board.set_pixel(x, y, "white")


# App ------------------

app = App("Destroy the dots")

instructions = Text(app, text="Click the dots to destroy them")
board = Waffle(app, width=GRID_SIZE, height=GRID_SIZE,
command=destroy_dot)
board.after(1000, add_dot)

app.display()
```

destroy5.py

Language: **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, Text, Waffle
004.  from random import randint
005.
006.  # Variables -------------
007.
008.  GRID_SIZE = 5
009.  score = 0
010.
011.  # Functions -------------
012.
013.  def add_dot():
014.      x, y = randint(0,GRID_SIZE-1), randint(0,GRID_SIZE-1)
015.      while board[x, y].dotty == True:
016.          x, y = randint(0,GRID_SIZE-1), randint(0,GRID_SIZE-1)
017.      board[x, y].dotty = True
018.      board.set_pixel(x, y, "red")
019.      board.after(1000, add_dot)
020.
021.  def destroy_dot(x,y):
022.      global score
023.      if board[x,y].dotty == True:
024.          board[x,y].dotty = False
025.          board.set_pixel(x, y, "white")
026.          score += 1
027.          score_display.value = "Your score is " + str(score)
028.
029.
030.  # App ------------------
031.
032.  app = App("Destroy the dots")
033.
034.  instructions = Text(app, text="Click the dots to destroy them")
035.  board = Waffle(
      app, width=GRID_SIZE, height=GRID_SIZE, command=destroy_dot)
036.  board.after(1000, add_dot)
037.  score_display = Text(app, text="Your score is " + str(score))
038.
039   app.display()
```

Figure 3



▲ **Figure 3** Every second, a new dot will appear

Solution: add a score

First, add a variable in your variables section:

```
score = 0
```

Next, add a new Text widget in the app section to display the score:

```
score_display = Text(app, text="Your score
is " + str(score))
```

Finally, add 1 to the score every time a dot is destroyed:

```
def destroy_dot(x,y):

    # Declare score global
    global score

    # This code already exists
    if board[x,y].dotty == True:
        board[x,y].dotty = False
        board.set_pixel(x, y, "white")

        # Add 1 to score and display it on
the GUI
        score += 1
        score_display.value = "Your score is
" + str(score)
```

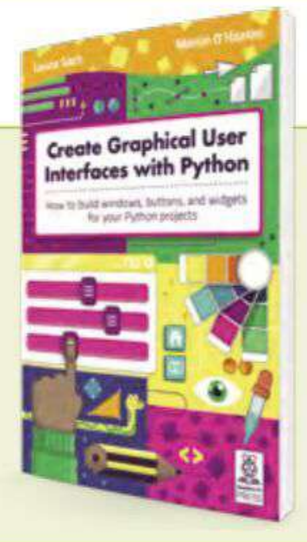
Your code (without the optional comments) should now resemble **destroy5.py**. Test your game and you should see your score go up by 1 every time you click on a dot.

Put the player under pressure

Now that you can track the player's score, you can use it to put the player under pressure and speed up the spawn of dots if they are doing well.

Remember that you used an after call inside the `add_dot` function to schedule another dot in 1000

Create Graphical User Interfaces with Python



For further tutorials on how to make your own GUIs with guizero, take a look at our book, *Create Graphical User Interfaces with Python*. Its 156 pages are packed with essential info and a range of exciting projects.
magpi.cc/pythongui

milliseconds (or 1 second)? Go back and find that line – you're going to change it a bit.

First, create a variable `speed` and set it to 1000. Then, instead of scheduling a call to add a dot after 1000 ms, schedule it to add a dot after speed milliseconds. This will have absolutely no effect on the game... yet. You are still scheduling the next call after 1000 ms, but that figure is now coming from the variable speed instead of being hard-coded as a magic number.

```
speed = 1000
board.after(speed, add_dot)
```

Now here's how you can ramp up the pressure. Between these two lines of code, you can add some code to set the speed of dots depending on the current score. Here is an example:

```
speed = 1000
if score > 30:
    speed = 200
elif score > 20:
    speed = 400
elif score > 10:
    speed = 500
board.after(speed, add_dot)
```

Here, you can see that if the player has got more than 10 points, the new dots will appear every 500 ms, if they have more than 20 points a dot will appear every 400 ms, and so on. This makes the game much harder the more points you have. Save your code – **destroy6.py** – and test the game to see the difference. You can alter the numbers or add more `elif` conditions if you want to increase the difficulty even further.

Game over

All that remains is to figure out when the player has lost the game; this happens when every square has turned into a red dot.

Remember that when you made Tic-tac-toe, you used nested loops to check whether all squares were filled and the game was a draw? You can use the same method here too, to loop through every square on the grid and check if it is a red dot. In your `add_dot` function, just before the call to `after`, add some code for a nested loop to loop through all squares on the board:

```
all_red = True
for x in range(GRID_SIZE):
    for y in range(GRID_SIZE):
```

> We need to figure out when the player has lost the game; this happens when every square has turned into a red dot

destroy6.py

Language: **Python 3**

```
001.  # Imports --------------
002.
003.  from guizero import App, Text, Waffle
004.  from random import randint
005.
006.  # Variables ------------
007.
008.  GRID_SIZE = 5
009.  score = 0
010.
011.  # Functions ------------
012.
013.  def add_dot():
014.      x, y = randint(0,GRID_SIZE-1), randint(0,GRID_SIZE-1)
015.      while board[x, y].dotty == True:
016.          x, y = randint(0,GRID_SIZE-1), randint(0,GRID_SIZE-1)
017.      board[x, y].dotty = True
018.      board.set_pixel(x, y, "red")
019.
020.      speed = 1000
021.      if score > 30:
022.          speed = 200
023.      elif score > 20:
024.          speed = 400
025.      elif score > 10:
026.          speed = 500
027.      board.after(speed, add_dot)
028.
029.  def destroy_dot(x,y):
030.      global score
031.      if board[x,y].dotty == True:
032.          board[x,y].dotty = False
033.          board.set_pixel(x, y, "white")
034.          score += 1
035.          score_display.value = "Your score is " + str(score)
036.
037.
038.  # App ------------------
039.
040.  app = App("Destroy the dots")
041.
042.  instructions = Text(app, text="Click the dots to destroy them")
043.  board = Waffle(
      app, width=GRID_SIZE, height=GRID_SIZE, command=destroy_dot)
044.  board.after(1000, add_dot)
045.  score_display = Text(app, text="Your score is " + str(score))
046.
047.  app.display()
```

07-destroy-the-dots.py

Language: **Python 3**

```
001.  # Imports --------------
002.
003.  from guizero import App, Text, Waffle
004.  from random import randint
005.
006.  # Variables ------------
007.
008.  GRID_SIZE = 5
009.  score = 0
010.
011.  # Functions ------------
012.
013.  def add_dot():
014.      x, y = randint(0,GRID_SIZE-1), randint(0,GRID_SIZE-1)
015.      while board[x, y].dotty == True:
016.          x, y = randint(0,GRID_SIZE-1), randint(0,GRID_SIZE-1)
017.      board[x, y].dotty = True
018.      board.set_pixel(x, y, "red")
019.
020.      speed = 1000
021.      if score > 30:
022.          speed = 200
023.      elif score > 20:
024.          speed = 400
025.      elif score > 10:
026.          speed = 500
027.
028.      all_red = True
029.      for x in range(GRID_SIZE):
030.          for y in range(GRID_SIZE):
031.              if board[x,y].color != "red":
032.                  all_red = False
033.      if all_red:
034.          score_display.value = "You lost! Score: " + str(score)
035.      else:
036.          board.after(speed, add_dot)
037.
038.  def destroy_dot(x,y):
039.      global score
040.      if board[x,y].dotty == True:
041.          board[x,y].dotty = False
042.          board.set_pixel(x, y, "white")
043.          score += 1
044.          score_display.value = "Your score is " + str(score)
045.
046.  # App ------------------
047.
048.  app = App("Destroy the dots")
049.
050.  instructions = Text(app, text="Click the dots to destroy them")
051.  board = Waffle(
      app, width=GRID_SIZE, height=GRID_SIZE, command=destroy_dot)
052.  board.after(1000, add_dot)
053.  score_display = Text(app, text="Your score is " + str(score))
054.
055.  app.display()
```

The first line begins by assuming that all squares are red. The nested loop will provide the co-ordinates of every square on the grid in turn, as the values `x` and `y`, so that you can check whether this is true.

Add some code inside the second loop to find out whether the current pixel is red, and if it is *not*, change the `all_red` variable to False.

```
all_red = True
for x in range(GRID_SIZE):
    for y in range(GRID_SIZE):
        if board[x,y].color != "red":
            all_red = False
```

> After both loops end, check whether the grid contains all red dots. If so, the player has lost

After both loops end (make sure you unindent the following code), check whether the grid contains all red dots. If so, the player has lost, so display a message:

```
if all_red:
    score_display.value = "You lost! Score:
" + str(score)
```

If the player hasn't lost, the game should continue. Add an `else:` and inside it, indent the `after` method you already have, as we only want to add a new dot if the player has not lost:

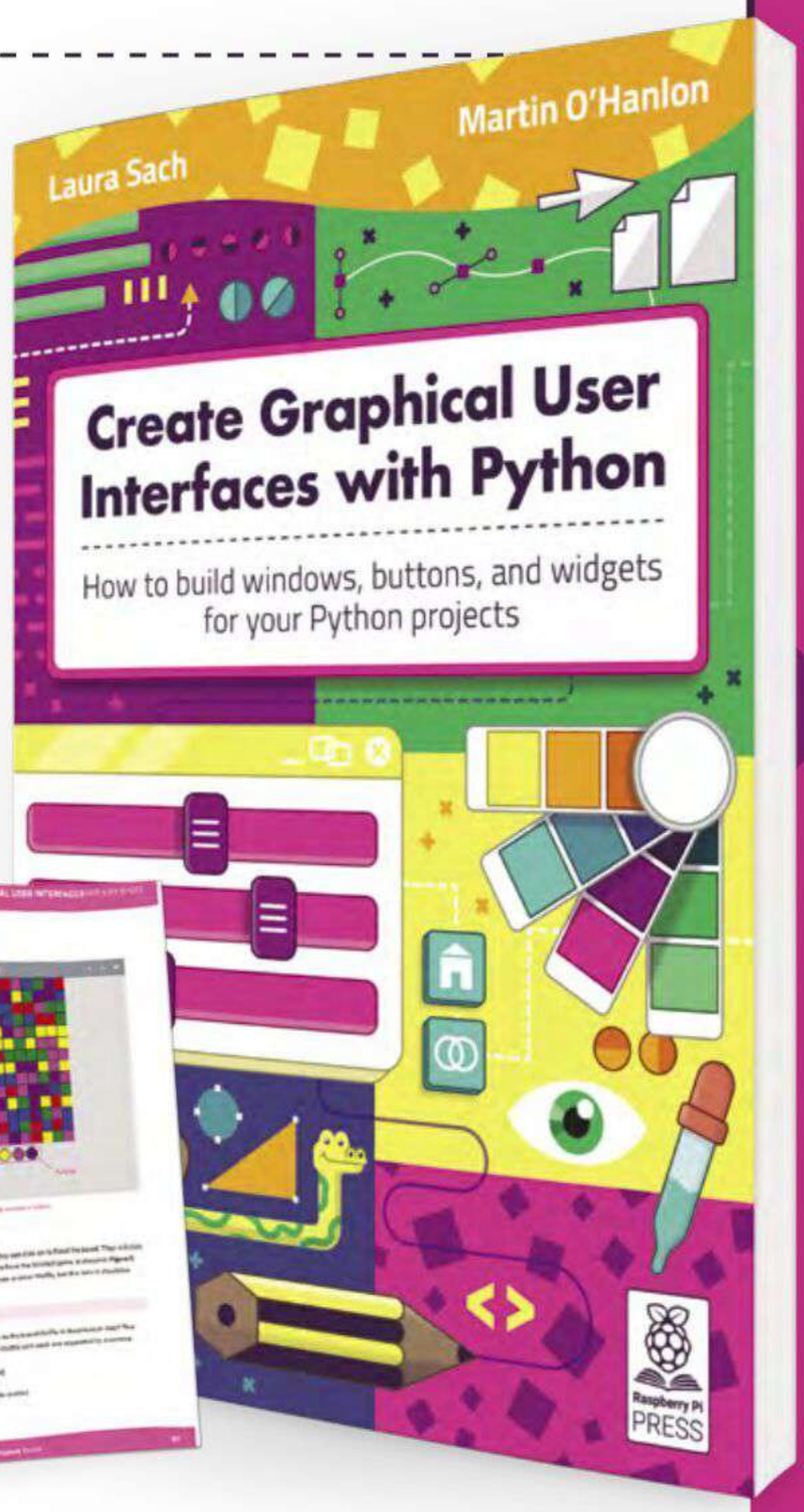
```
else:
    board.after(speed, add_dot)
```

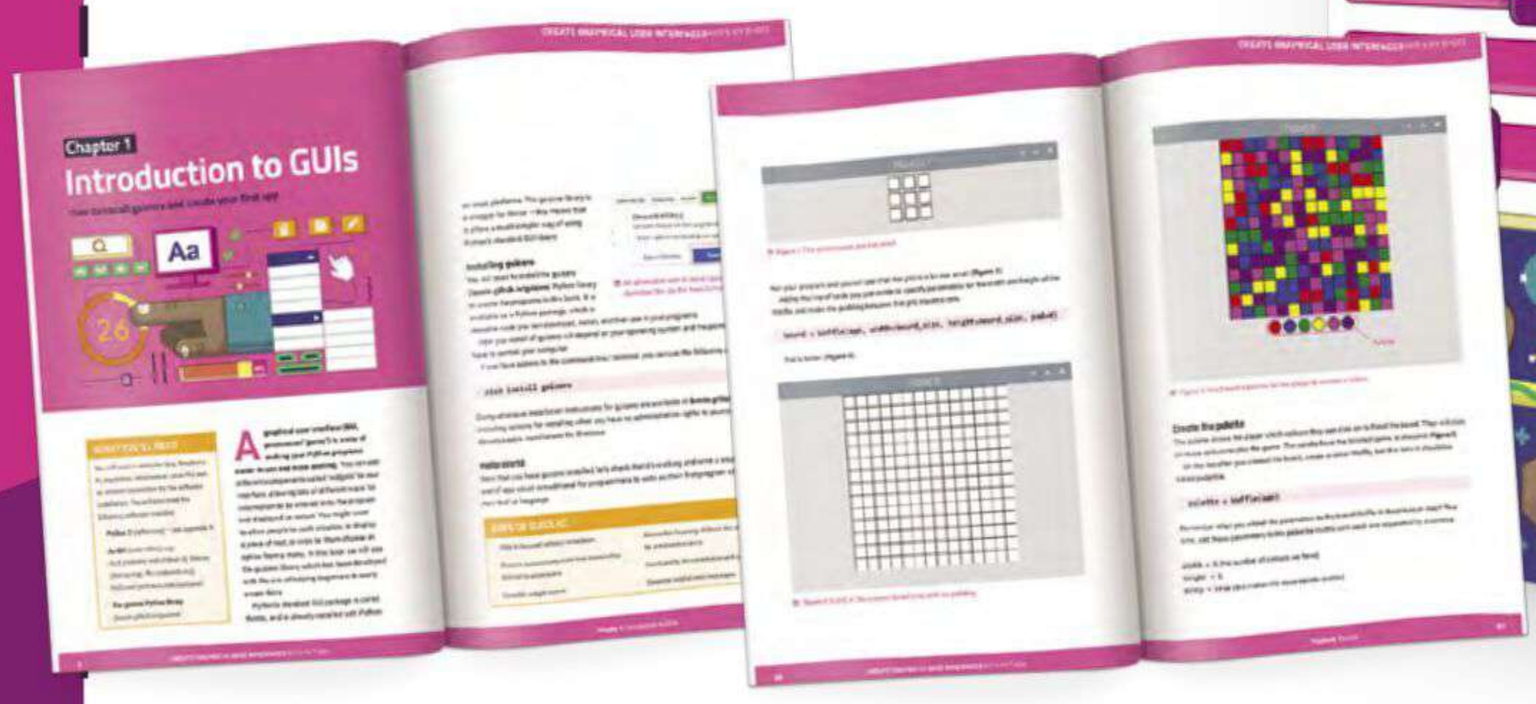
Be careful to indent the after line you already have here rather than adding another one, or your game will start behaving strangely and generate multiple dots per second!

Your final code should resemble that in **07-destroy-the-dots.py**. Enjoy the game!

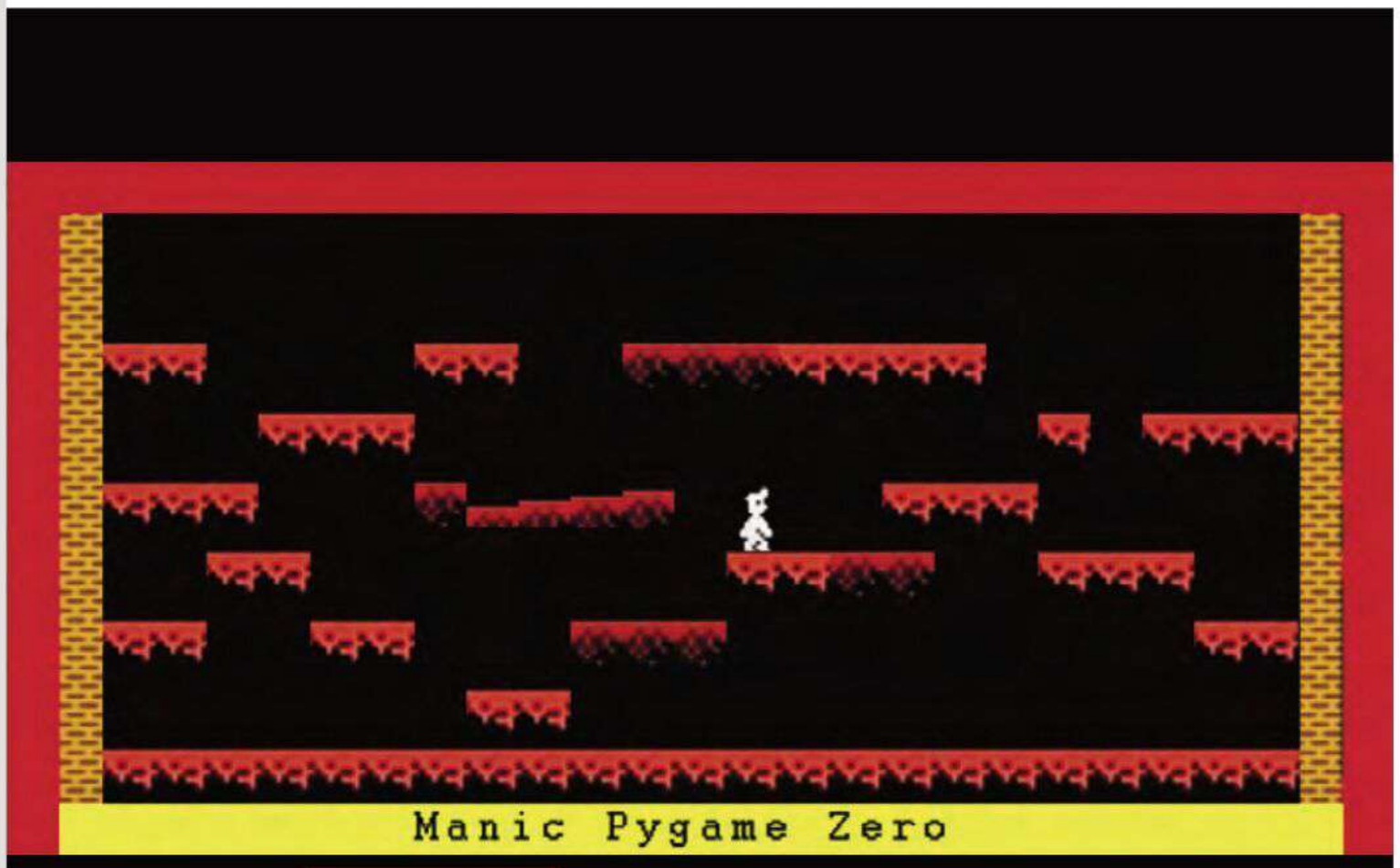
Create Graphical User Interfaces with Python

- ✓ **Create games and fun Python programs**
- ✓ Learn how to create your own graphical user interfaces
- ✓ **Use windows, text boxes, buttons, images, and more**
- ✓ Learn about event-based programming
- ✓ **Explore good (and bad) user interface design**





Buy online: magpi.cc/pythongui



Source Code

Wireframe

This tutorial first appeared in Wireframe, our sister magazine that lifts the lid on the world of video games. Every issue includes tutorials and in-depth interviews, along with news and reviews of the latest indie and triple-A games.

To find out more, visit their website at **wfmag.cc**.

Check out their subscription offers at **wfmag.cc/subscribe**.

Remake Manic Miner's collapsing platforms

Traverse a crumbly cavern in our homage to a Spectrum classic

AUTHOR
MARK VANSTONE

One of the most iconic games on the Sinclair ZX Spectrum featured a little man called Miner Willy, who spent his days walking and jumping from platform to platform collecting the items needed to unlock the door on each screen. *Manic Miner*'s underground world featured caverns, processing plants, killer telephones, and even a forest featuring little critters that looked suspiciously like Ewoks.

Written by programmer Matthew Smith and released by Bug-Byte in 1983, the game became one of the most successful titles on the Spectrum. Smith was only 16 when he wrote *Manic Miner* and even constructed his own hardware to speed up the development process, assembling the code on a TRS-80 and then downloading it to the Spectrum with his own hand-built interface. The success of *Manic Miner* was then closely followed by *Jet Set Willy*, featuring the same character, and although they were originally written for the Spectrum, the games very soon made it onto just about every home computer of the time.

Both *Manic Miner* and *Jet Set Willy* featured unstable platforms which crumbled in Willy's wake, and it's these we're going to try to recreate this month.

In this Pygame Zero example, we need three frames of animation for each of the two directions of movement. As we press the arrow keys we can move the Actor left and right, and in this case we'll decide which frame to display based on a `count` variable which is incremented each time our `update()` function runs. We can create platforms from a two-dimensional data list representing positions on the screen, with 0 meaning a blank space, 1 being a solid platform, and 2 a collapsible platform. To set these up, we run through the list and make Actor objects for each platform segment.

For our `draw()` function, we can blit a background graphic, then Miner Willy, and then our platform blocks. During our `update()` function, apart from checking key presses, we also need to do some gravity calculations. This will mean that if Willy isn't standing on a platform or jumping, he'll start to fall towards the bottom of the screen. Instead of checking to see if Willy has collided with the whole platform, we only check to see if his feet are in contact with the top. This means he can jump up through the platforms but will then land on the top and stop. We set a variable to indicate that Willy's standing on the ground so that when the **SPACE** bar is pressed, we know if he can jump or not. While we're checking if Willy's on a platform, we also check to see if it's a collapsible one, and if so, we start a timer so that the platform moves downwards and eventually disappears. Once it's gone, Willy will fall through. The reason we have a delayed timer rather than just starting the platform heading straight down is so that Willy can run across many tiles before they collapse, but his way back will quickly disappear. The disappearing platforms are achieved by changing the image of the platform block as it moves downward.

As we've seen, there were several other elements to each *Manic Miner* screen, such as roaming bears that definitely weren't from *Star Wars*, and those dastardly killer telephones. We'll leave you to add those...

Crumbly platforms in Python

Here's Mark's code for a *Manic Miner* screen, complete with collapsing platforms. To get it working on your system, you'll need to install Pygame Zero – full instructions are available at **wfmag.cc/pgzero**.

Download the code from GitHub: **wfmag.cc/wfmag49**

```
# Manic Miner

HEIGHT = 400
willy = Actor('willyr0',(400,300))
willy.direction = "r"
willy.jump = 0
willy.onground = False
count = 0
platforms = [[1,1,0,0,0,0,1,1,0,0,2,2,2,1,1,1,1,0,0,0,0,0,0],
             [0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1],
             [1,1,1,0,0,0,2,2,2,2,2,0,0,0,0,1,1,1,0,0,0,0,0],
             [0,0,1,1,0,0,0,0,0,0,0,0,1,1,2,2,0,0,1,1,1,0,0],
             [1,1,0,0,1,1,0,0,0,2,2,2,0,0,0,0,0,0,0,0,0,1,1],
             [0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0]]
platformActors = []
for r in range(len(platforms)):
    for c in range(len(platforms[r])):
        if(platforms[r][c] != 0 ): platformActors.
append(Actor('platform'+str(platforms[r][c])+"1",(70+(c*30),1
20+(r*40))))
        platformActors[len(platformActors)-1].status = 0

def draw():
    screen.blit("background", (0, 0))
    willy.draw()
    drawPlatforms()

def update():
    global count
    willy.image = "willy"+ willy.direction + "0"
    if keyboard.left:
        moveWilly(-1,0)
        willy.direction = "l"
        willy.image = "willyl"+ str(int(count/8)%3)
        pass
    if keyboard.right:
        moveWilly(1,0)
        willy.direction = "r"
        willy.image = "willyr"+ str(int(count/8)%3)
        pass
    checkGravity()
    count += 1

def on_key_down(key):
    if key.name == "SPACE":
        if willy.onground == True:
            willy.jump = 40

def drawPlatforms():
    for p in range(len(platformActors)):
        if platformActors[p].status != -1:
            platformActors[p].draw()

def moveWilly(x,y):
    if willy.x+x < 730 and willy.x+x > 70:
        willy.x += x

def checkGravity():
    if willy.jump > 0:
        willy.y -=2
        willy.jump -=1
    if willy.y < 320:
        willy.onground = False
        for p in range(len(platformActors)):
            frame = int(platformActors[p].image[-1])+1
            if platformActors[p].status > 0 :
                platformActors[p].status -= 1
                if platformActors[p].status == 0 :
                    platformActors[p].y += 1
                    if frame > 8 :
                        platformActors[p].status = -1
                    else:
                        platformActors[p].image =
"platform2"+str(frame)
                        platformActors[p].status = 30
            if((willy.x > platformActors[p].x-20 and willy.x <
platformActors[p].x+20) and willy.y+20 == platformActors[p].y-
14+(frame-1) and platformActors[p].status != -1):
                willy.onground = True
                if platformActors[p].image[8] == "2":
                    if platformActors[p].status == 0 :
platformActors[p].status = 30
        if willy.onground == False:
            willy.y += 1
    else:
        willy.onground = True
```

Miner Willy makes his way to the exit, avoiding those vicious eighties telephones.

Raspberry Pi Pico burglar alarm

Build a handy alarm system to protect you and yours, using a Pico and some passive infrared sensors

MAKER

Gareth Halfacree

With a passion for open-source software and hardware, Gareth was an early adopter of the Raspberry Pi platform and has written several publications on its capabilities and flexibility.

@ghalfacree

You'll Need

- Raspberry Pi Pico
- Breadboard
- LED (any colour)
- 330 Ω resistor
- Piezoelectric buzzer
- HC-SR501 passive infrared (PIR sensors
- Male-to-male (M2M) and male-to-female (M2F) jumper wires

Another real-world use of microcontrollers is in alarm systems. From the alarm clock that gets you up in the morning to fire alarms, burglar alarms, and even the alarms that sound when there's a problem at a nuclear power station, microcontrollers help keep us all safe.

In this guide you'll build your own burglar alarm, which works like a commercial version: a special motion sensor keeps watch for anyone entering the room when they shouldn't be there, and flashes a light while sounding a siren to alert people to the intrusion. Whether you're protecting a bank vault or just trying to keep spying siblings out of your room, a burglar alarm is sure to come in handy.

The HC-SR501 PIR sensor

In our previous Pico tutorials, you've been working with simple input components in the form of push-button switches. This time, you're going to be using a specialised input known as a passive infrared sensor or PIR. There are hundreds of different PIR sensors available; the HC-SR501 is low-cost, high-performance, and works perfectly with your Pico.

If you've picked up a different model of sensor, look at its documentation to double-check which pins are which; also make sure that it operates at a 3V3 logic level, just like your Pico – if you connect a sensor which uses a higher voltage, such as a 12 V sensor, it will damage your Pico beyond repair. Some sensors may need a small switch or a jumper changing to move between logic voltage levels; this will be noted in the documentation.

A passive infrared sensor is designed to detect movement, in particular movement from people and other living things. It works a little like a camera, but instead of capturing visible light it looks for the heat emitted from a living body as infrared radiation. It's known as a passive infrared sensor, rather than active, because just like a camera sensor it doesn't send out any infrared signals of its own.

The actual sensor is buried underneath a plastic lens, typically shaped like a half-ball. The lens isn't technically necessary for the sensor to work, but serves to provide a wider field of vision (FOV); without the lens, the PIR sensor would only be able to see movement in a very narrow angle directly in front of the sensor. The lens serves to pull in infrared from a much wider angle, meaning a single PIR sensor is able to watch for movement over the majority of a room.

"A passive infrared sensor is designed to detect movement"

In commercial burglar alarm systems, a PIR sensor is only one of the sensors used; others include break-glass sensors which can tell when a window has been smashed, magnetic sensors which monitor whether a door is open or closed, acoustic sensors which can pick up a burglar's footsteps, and vibration sensors for telling if a lock is being forced open. A simple PIR sensor, though, is often enough for a low-security area – think a reception room, rather than a bank vault.

Pick up your HC-SR501 sensor now and take a look at it. The first thing to notice is that it has a circuit board of its own, a lot like your Pico – only smaller. As well as the sensor and lens, there are several other components: a small black integrated circuit (IC) which drives the sensor, some capacitors, and tiny surface-mount resistors. You may also see some small potentiometers which you can twist with a screwdriver to adjust the sensitivity of the sensor and how long it stays active when triggered; leave these as they are for now.

You'll also see three male pins, exactly like the pins on the bottom of your Pico. You can't push these directly into your breadboard, though, as the components on the board will get in the way. Instead, take three male-to-female (M2F) jumper

DOWNLOAD THE FULL CODE:

magpi.cc/github

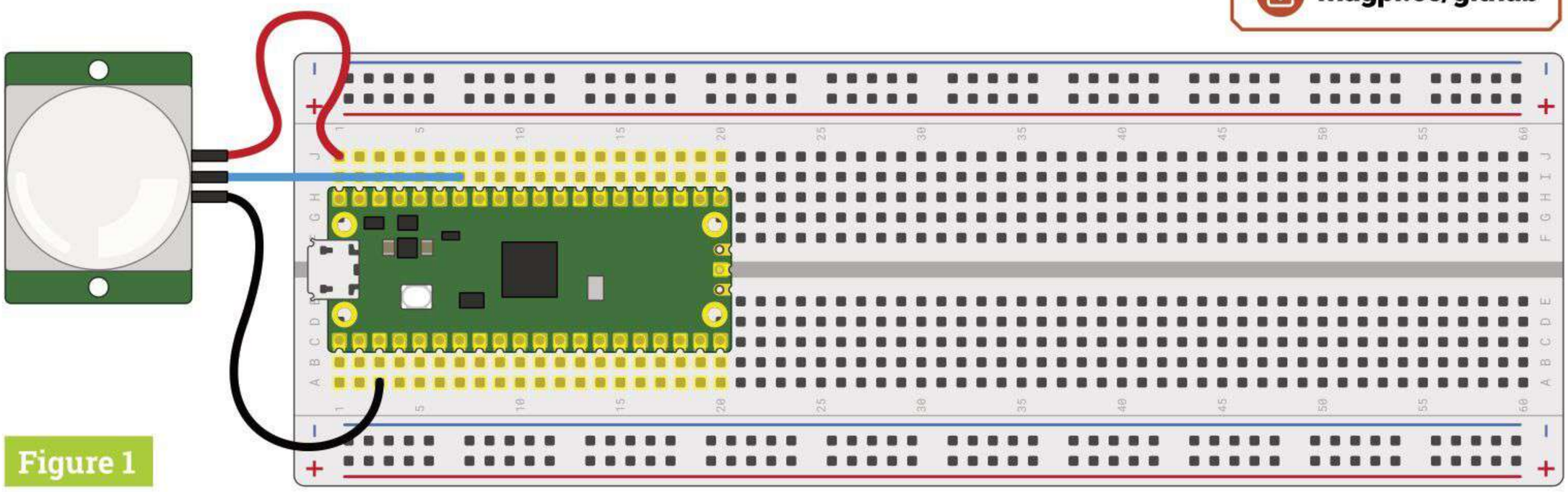


Figure 1

wires and insert the female ends onto the pins on your HC-SR501.

Next, take the male ends and wire them to the breadboard and your Pico. You'll need to check the documentation for your sensor when wiring it to your Pico: a lot of different companies make HC-SR501 sensors, and they don't always use the same pins for the same purposes. For the sensor illustrated in **Figure 1**, the pins are set up so that the ground (GND) pin is on the bottom, the signal or trigger pin is in the middle, and the power pin is on the right; your sensor may need the wires placing in a different order!

Start with the ground wire: this needs to be connected to any of your Pico's ground pins. In **Figure 1**, it's connected to the first ground pin in breadboard row 3. Next, connect the signal wire: this should be connected to your Pico's GPIO pin GP28.

Finally, you need to connect the power wire. Don't connect this to your Pico's 3V3 pin, though: the HC-SR501 is a 5V device, meaning that it needs 5 volts of electricity in order to work. If you wire the sensor to your Pico's 3V3 pin, it won't work – the pin simply doesn't provide enough power.

To give your sensor the 5V power it needs, wire it to the very top-right pin of your Pico – VBUS. This pin is connected to the micro USB port on your Pico, and taps into the USB 5V power line before it's converted to 3.3V to run your Pico's microprocessor. All three HC-SR501 pins should now be wired to your Pico: ground, signal, and power.

Programming your alarm

You'll need to program your Pico to recognise the sensor. Handily, this is no more difficult than reading a button – in fact, you can use the very same code. Start by creating a new program and importing the machine library so you can configure your Pico's GPIO pin, along with the utime library which we'll use to set delays in the program:

```
import machine
import utime
```

Then set up the pin you wired your HC-SR501 sensor to, GP28:

```
sensor_pir = machine.Pin(28, machine.Pin.IN,
machine.Pin.PULL_DOWN)
```

Like the reaction game in issue 105 (**magpi.cc/105**), a burglar alarm's inputs should act as an interrupt – stop the program doing whatever it was doing and react whenever the sensor is triggered. Start by defining a callback function to handle the interrupt:

```
def pir_handler(pin):
    utime.sleep_ms(100)
    if pin.value():
        print("ALARM! Motion detected!")
```

Make sure you have correct indentation. The `utime.sleep_ms(100)` and `if pin.value():` lines are used to prevent the alarm from being triggered by any jitter in the signal from the PIR sensor. The process of smoothing out fluctuations, as we've done here, is known as debouncing.

Finally, set up the interrupt itself. Remember that this isn't part of the handler function, so you'll need to delete any extra spaces Thonny automatically inserts:

▲ **Figure 1** Wiring an HC-SR501 PIR sensor to your Pico

Top Tip

Narrowing the FOV

PIR sensors are designed to cover as wide a field of vision (FOV) as possible so that burglars can't simply scoot around the edges of a room. If you find your sensor is triggering when you don't want it to, there's a simple way to fix it: get the cardboard inner tube from a toilet roll and place the sensor at the bottom. The tube will act like horse blinkers, stopping the sensor from seeing things to the side so it can concentrate on things further ahead.

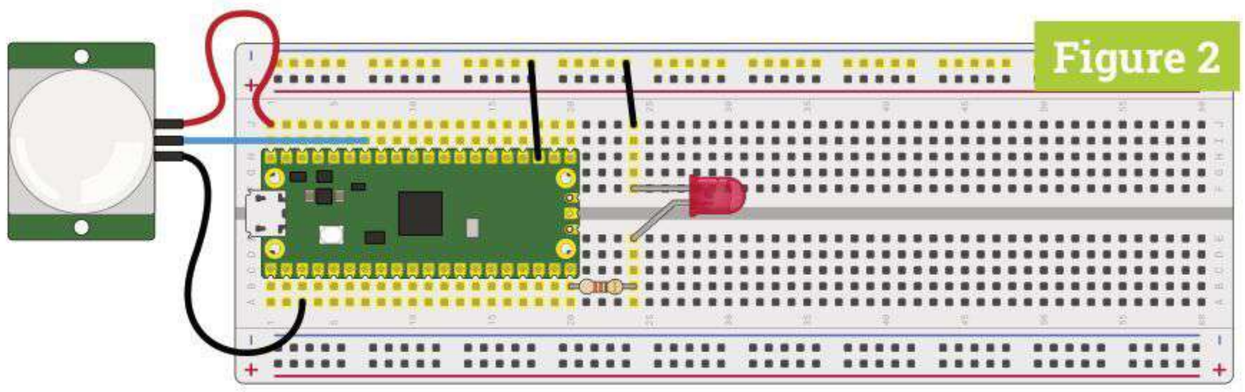


Figure 2

▲ **Figure 2** Adding an LED to the burglar alarm

```
sensor_pir.irq(trigger=machine.Pin.IRQ_RISING, handler=pir_handler)
```

That's enough for the moment: remember that interrupts stay active regardless of what the rest of the program is doing, so there's no need to add an infinite loop to keep your program running. Click the Run icon and save the program to your Pico as **Burglar_Alarm.py**.

Wave your hand over the PIR sensor: a message will print to the Shell area confirming that the sensor saw you. If you keep waving your hand, the message will keep printing – but with a delay between each time it's printed.

That delay isn't part of your program, but built into the HC-SR501 hardware itself: the sensor sends the trigger signal to your Pico's GPIO pin when motion is detected, and keeps that signal switched on for several seconds before dropping it. On most HC-SR501 sensors, that delay is adjusted using one of the small potentiometers: you can insert a screwdriver and turn it one way to decrease the delay and the other way to increase the delay. Check your sensor's documentation for which potentiometer controls the delay.

Because your interrupt trigger is set to fire on the rising edge of the signal, the message is printed as soon as the PIR sensor sends its signal of 1 or 'high'. Even if more motion is detected, the interrupt won't fire again until the built-in delay has passed and the signal has returned to 0, or 'low'.

Printing a message to the Shell is enough to prove your sensor is working, but it doesn't make for much of an alarm. Real burglar alarms have lights and sirens that alert everyone around that something's wrong – and you can add the same to your own alarm.

Start by wiring an LED, of any colour, to your Pico as shown in **Figure 2**. The longer leg, the anode, needs to connect to pin GP15 via a 330 Ω resistor – remember that without this resistor in place to limit the amount of current passing through the LED, you can damage both the LED and your Pico. The shorter leg, the cathode, needs to be wired to one of your Pico's ground pins – use your breadboard's ground rail and two male-to-male (M2M) jumper wires for this.

Top Tip

Value vs toggle

There are times when using the toggle function over setting a value makes sense, like when blinking an LED – but make sure you've thought through what you're trying to achieve first. If your project hinges on an output definitely being on or off at a given time – such as a warning light, or a pump which drains a water tank – always explicitly set the value rather than relying on a toggle.

To set up the LED output, add a new line just below where you set up the PIR sensor's pin:

```
led = machine.Pin(15, machine.Pin.OUT)
```

That's enough to configure the LED, but you'll need to make it light up. Add the following new line to your interrupt handler function, under the `print` line (it should be indented by eight spaces to match that) :

```
for i in range(50):
```

You've just created a finite loop, one which will run 50 times. The letter `i` represents an increment, a value which goes up each time the loop runs, and which is populated by the instruction `range(50)`.

Give your new loop something to do, remembering that these lines underneath will need to be indented by a further four spaces (twelve in all), as they form both part of the loop you just opened and the interrupt handler function:

```
led.toggle()
utime.sleep_ms(100)
```

The first of these lines is a feature of the machine library which lets you simply change the value of an output pin, rather than set a value – so if the pin is currently 1, or high, toggling it will set it to 0, or low; if the pin is already 0, or low, toggling it will set it to high. These two lines of code flash the LED on and off with a 100-millisecond – a tenth of a second – delay. The result is similar to the LED blinking program we made in issue 102 (**magpi.cc/102**).

Your program will now look like this:

```
import machine
import utime

sensor_pir = machine.Pin(28, machine.Pin.IN, machine.Pin.PULL_DOWN)
led = machine.Pin(15, machine.Pin.OUT)

def pir_handler(pin):
    utime.sleep_ms(100)
    if pin.value():
        print("ALARM! Motion detected!")
        for i in range(50):
            led.toggle()
            utime.sleep_ms(100)

sensor_pir.irq(trigger=machine.Pin.IRQ_RISING, handler=pir_handler)
```

Click the Run button, then wave your hand over the PIR sensor again: you'll see the usual alarm message print to the Shell area, then the LED will begin rapidly flashing as a visual alert. Wait for the LED to stop flashing, then wave your hand over the PIR sensor again: the message will print again, and the LED will repeat its flashing pattern.

To make your burglar alarm even more of a deterrent, you can make it flash slowly even when there's no motion being detected – warning would-be intruders that your room is under observation. Go to the very bottom of your program and add in the following lines:

```
while True:
    led.toggle()
    utime.sleep(5)
```

Click Run again, but leave the PIR sensor alone: you'll see the LED is now turning on for five seconds, then turning off for five seconds. This pattern will continue as long as the sensor isn't triggered; wave your hand over the PIR sensor and you'll see the LED rapidly flashing again, before going back to its slow-flash pattern.

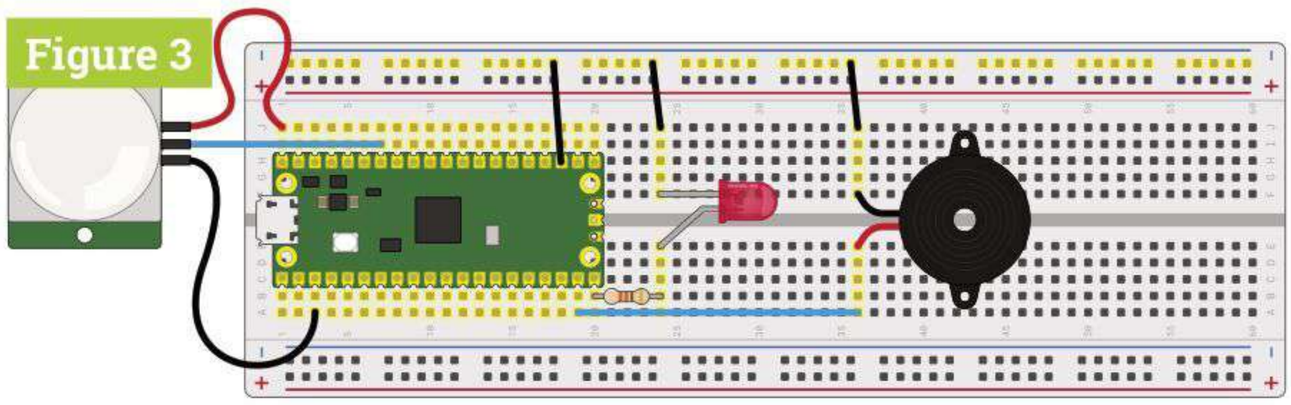
This highlights a key difference between threads and interrupts: if you'd written this program using threads, your program would still be trying to toggle the LED on and off on a five-second interval even while your PIR handler is flashing it on and off on a 100-millisecond interval. That's because threads run concurrently, side by side.

With interrupts, the main program is paused while the interrupt handler runs – so the five-second toggle code you've written stops until the handler has finished flashing the LED, then picks up from where it left off. Whether you need your code to pause or to carry on running is the key to whether you need to use threads or interrupts, and will depend on exactly what your project is trying to do.

Inputs and outputs: putting it all together

Your burglar alarm now has a flashing LED to warn intruders away, and a way to see when it's been triggered without having to watch the Shell area for a message. Now all it needs is a siren – or, at least, a piezoelectric buzzer, which makes sound without deafening your neighbours.

Depending on which model you purchased, your piezoelectric buzzer will have either pins sticking out of the bottom or short wires attached to its sides. If the buzzer has pins, insert these into your breadboard so the buzzer is straddling the centre divide; if it has wires, place these in the breadboard and simply rest the buzzer on the breadboard.



▲ **Figure 3** Wiring a two-wire piezoelectric buzzer

If your buzzer has long enough wires, you might be able to connect them to the breadboard right next to your Pico's GPIO pins; if not, use male-to-male (M2M) jumper wires to wire the buzzer as shown in **Figure 3**. The red wire, or the positive pin marked with a + symbol, should be connected to pin GP14 at the bottom-left of your Pico, just above the pin you're using for the LED. The black wire, or the negative pin marked with a minus (-) symbol or the letters GND, needs to be connected to the ground rail of your breadboard.

If your buzzer has three pins, connect the leg marked with a minus symbol (-) or the letters GND to the ground rail of your breadboard, the pin marked with S or SIGNAL to pin GP14 on your Pico, and the remaining leg – which is usually the middle leg – to the 3V3 pin on your Pico.

If you run your program now, nothing will change: the buzzer will only make a sound when it receives power from your Pico's GPIO pins. Go back to the top of your program and set the buzzer up just below where you set the LED up:

```
buzzer = machine.Pin(14, machine.Pin.OUT)
```

Next, change your interrupt handler to add a new line below `led.toggle()` – remembering to indent it by twelve spaces to match:

```
buzzer.toggle()
```

Your program will now look like this:

```
import machine
import utime

sensor_pir = machine.Pin(28, machine.Pin.IN,
machine.Pin.PULL_DOWN)
led = machine.Pin(15, machine.Pin.OUT)
buzzer = machine.Pin(14, machine.Pin.OUT)

def pir_handler(pin):
    utime.sleep_ms(100)
    if pin.value():
        print("ALARM! Motion detected!")
```

Warning!

When using an active buzzer, it will continue to sound for as long as the pin it's connected to is high – in other words, has a value of 1. Because your loop runs an even number of times, it finishes with the buzzer switched off; change the loop to run an odd number of times, though, and it will finish with the buzzer still sounding – and pressing the Stop button won't turn it off. If this happens, simply unplug your Pico's micro USB cable and plug it back in again – then change your program so it doesn't happen again!

```
            for i in range(50):
                led.toggle()
                buzzer.toggle()
                utime.sleep_ms(100)

sensor_pir.irq(trigger=machine.Pin.IRQ_
RISING, handler=pir_handler)

while True:
    led.toggle()
    utime.sleep(5)
```

Click Run and wave your hand over the PIR sensor: the LED will flash rapidly, as before, but this time it'll be accompanied by a beeping sound from the buzzer. Congratulations: that should be more than enough to scare an intruder away from ransacking your secret stash of sweets!

If you find your buzzer is clicking, rather than beeping, then you're using a passive buzzer rather than an active buzzer. An active buzzer has a component inside known as an oscillator, which rapidly moves the metal plate to make the buzzing sound; a passive buzzer lacks this component, meaning you need to replace it with some code of your own.

If you're using a passive buzzer, try this version of the program instead – it toggles the pin connected to the buzzer on and off very quickly, mimicking the effect of the oscillator in an active buzzer:

```
import machine
import utime

sensor_pir = machine.Pin(28, machine.Pin.IN,
machine.Pin.PULL_DOWN)
led = machine.Pin(15, machine.Pin.OUT)
buzzer = machine.Pin(14, machine.Pin.OUT)

def pir_handler(pin):
    utime.sleep_ms(100)
    if pin.value():
        print("ALARM! Motion detected!")
        for i in range(50):
            led.toggle()
            for j in range(25):
                buzzer.toggle()
                utime.sleep_ms(3)

sensor_pir.irq(trigger=machine.Pin.IRQ_
RISING, handler=pir_handler)

while True:
    led.toggle()
    utime.sleep(5)
```

Note that the new loop, which controls the buzzer, doesn't use the letter `i` to track the increment; that's because you're already using that letter for the outer loop – so it uses the letter `j`. The short delay of just three milliseconds, meanwhile, means the pin connected to the buzzer turns on and off rapidly enough for it to make a buzzing noise.

Try changing the delay to four milliseconds instead of three and you'll find the buzzer sounds at a lower pitch. Changing the delay changes the frequency of the buzzer's oscillation: a longer delay means it oscillates at a lower frequency, making it a lower-pitched sound; a shorter delay makes it oscillate at a higher frequency, making it a higher-pitched sound.

Extending your alarm

Burglar alarms rarely cover a single room: instead, they use a network of multiple sensors to monitor multiple rooms from a single alarm system. Your Pico-based burglar alarm can work in exactly the same way, adding in multiple sensors to cover multiple areas at once.

You'll need an HC-SR501 sensor for each area you want to cover; in this example you'll add one more sensor for a total of two, but you can keep going and add as many sensors as you need.

Both of your sensors need 5 V power to work, but you've already used the VUSB pin on your Pico for the first sensor. If your breadboard has enough room, you could put a male-to-female (M2F) jumper wire next to the one connected to your first sensor and use it for the second; a neater approach, though, is to use the power rail on your breadboard.

Disconnect the first sensor's power wire from the breadboard end, and insert it into the power rail coloured red or marked with a plus (+) symbol. Take a male-to-male (M2M) jumper wire and connect the same power rail to your Pico's VUSB pin. Next, take a male-to-female (M2F) jumper wire and connect the power rail to your second PIR sensor's power input pin.

Finally, wire up the ground pin and signal pin of your second PIR sensor as before – but this time connect the signal pin to pin GP22 on your Pico, as shown in **Figure 4**. Your circuit now has two sensors, each connected to a separate pin. Setting your program up to read the second sensor as well as the first is as simple as adding two new lines. Start by setting the second sensor up, adding a new line below where you set the first sensor up:

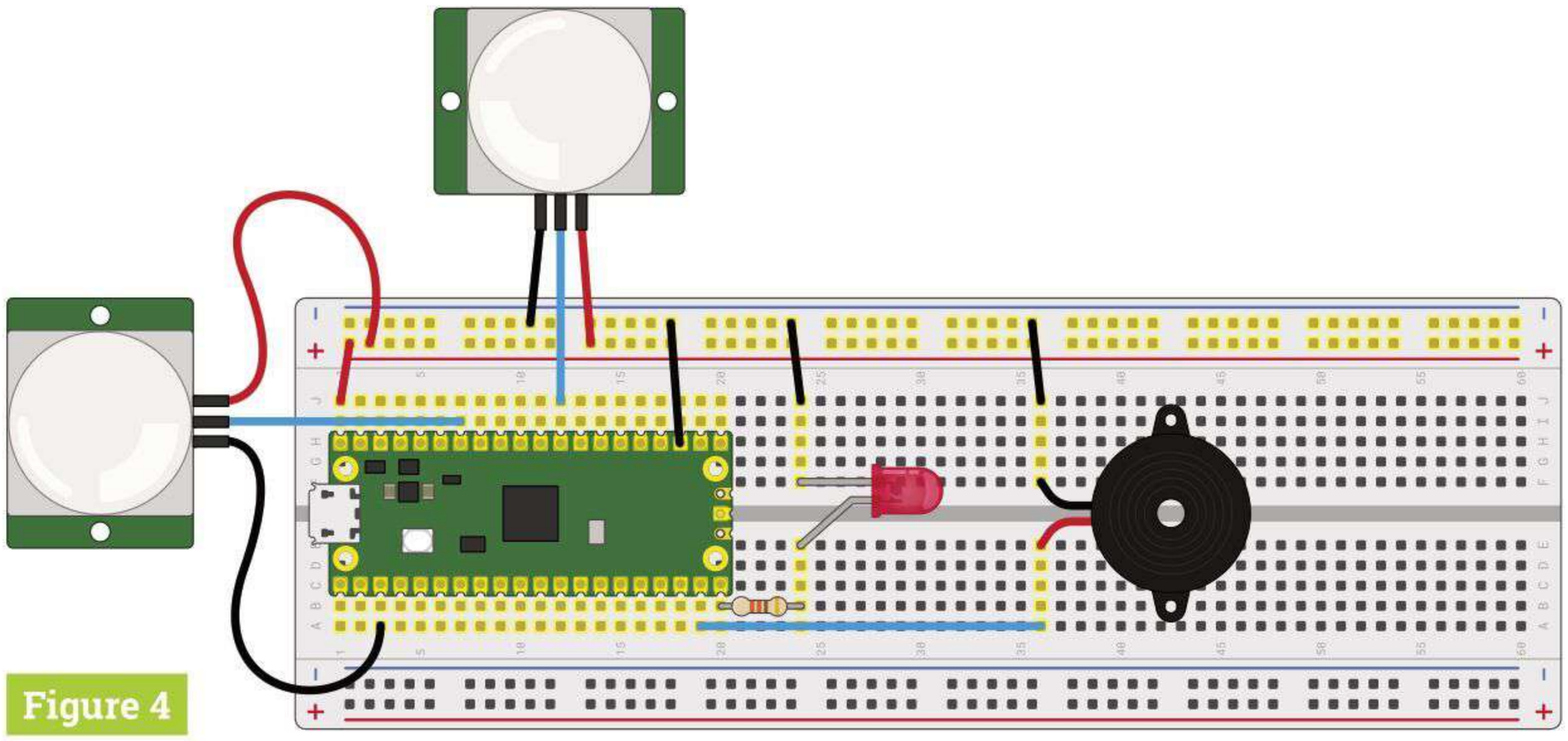


Figure 4

Figure 4 Adding a second PIR sensor to cover another room

```
  sensor_pir2 = machine.Pin(22, machine.Pin.IN,
machine.Pin.PULL_DOWN)
```

Then create a new interrupt, again directly beneath your first interrupt:

```
  sensor_pir2.irq(trigger=machine.Pin.IRQ_
RISING, handler=pir_handler)
```

Remember that you can have multiple interrupts with a single handler, so there's no need to change that part of your program.

Click Run, and wave your hand over the first PIR sensor: you'll see the alert message, the LED flash, and the buzzer sound as normal. Wait for them to finish, then wave your hand over the second PIR sensor: you'll see your burglar alarm respond in exactly the same way.

To make your alarm really smart, you can customise the message depending on which pin was responsible for the interrupt – and it works exactly the same way as in the two-player reaction game you wrote earlier. Go back to your interrupt handler and modify it so it looks like:

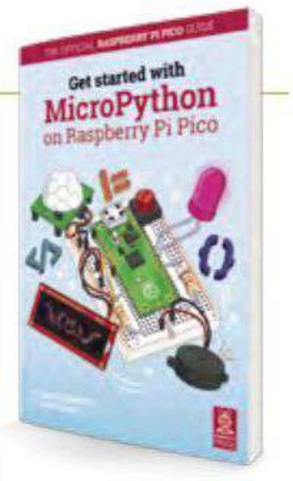
```
  def pir_handler(pin):
      utime.sleep_ms(100)
      if pin.value():
          if pin is sensor_pir:
              print("ALARM! Motion detected in
bedroom!")
          elif pin is sensor_pir2:
              print("ALARM! Motion detected in
living room!")
          for i in range(50):
              led.toggle()
              buzzer.toggle()
              utime.sleep_ms(100)
```

Just as in the reaction game project in issue 105, this code uses the fact that an interrupt reports which pin it was triggered by: if the PIR sensor attached to pin GP28 is responsible, it will print one message; if it was the PIR sensor attached to pin GP22, it will print another.

Your finished program will look like this:

```
  import machine
  import utime

  sensor_pir = machine.Pin(28, machine.Pin.IN,
machine.Pin.PULL_DOWN)
  sensor_pir2 = machine.Pin(22, machine.Pin.IN,
machine.Pin.PULL_DOWN)
  led = machine.Pin(15, machine.Pin.OUT)
  buzzer = machine.Pin(14, machine.Pin.OUT)

  def pir_handler(pin):
      utime.sleep_ms(100)
      if pin.value():
          if pin is sensor_pir:
              print("ALARM! Motion detected in
bedroom!")
          elif pin is sensor_pir2:
              print("ALARM! Motion detected in
living room!")
          for i in range(50):
              led.toggle()
              buzzer.toggle()
              utime.sleep_ms(100)

  sensor_pir.irq(trigger=machine.Pin.IRQ_RISING,
handler=pir_handler)
  sensor_pir2.irq(trigger=machine.Pin.IRQ_
RISING, handler=pir_handler)

  while True:
      led.toggle()
      utime.sleep(5)
```

If using a passive buzzer, you'll need to change the buzzer toggle to a loop to have it beep. Click Run and wave your hand over one sensor, then the other, to see both messages print to the Shell area. Congratulations: you now know how to build a burglar alarm that can cover multiple areas!



Get Started with MicroPython on Raspberry Pi Pico

For more physical computing projects to try on your Raspberry Pi Pico, grab a copy of the new book, *Get Started with MicroPython on Raspberry Pi Pico*. As well as learning how to use Raspberry Pi Pico's pins as inputs and outputs, you'll build a simple game, measure temperatures, save and load data to your Pico's file system, and even make a burglar alarm for your room. *Get Started with MicroPython on Raspberry Pi Pico* is available now from **magpi.cc/picobook**.

Part 01

Pico-Voice

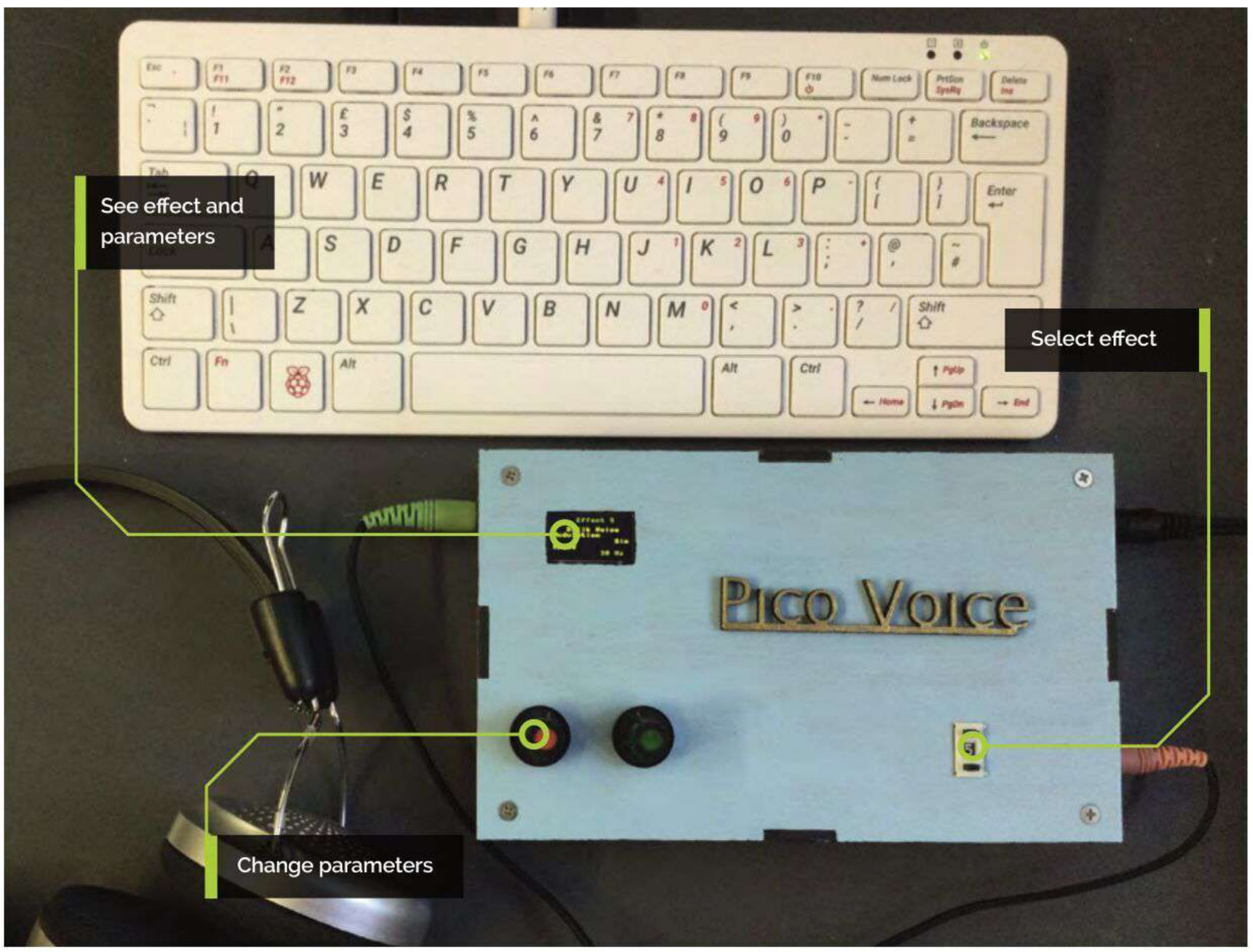
Use a Raspberry Pi Pico to make your own voice processing and sound effects system

MAKER

Mike Cook

Veteran magazine author from the old days, writer of the Body Build series, plus co-author of *Raspberry Pi for Dummies*, *Raspberry Pi Projects*, and *Raspberry Pi Projects for Dummies*.

magpi.cc/mikecook



You'll Need

- BCD switch **magpi.cc/bcdswitch**
- 2 × 3mm stereo jack sockets **magpi.cc/stereojack**
- 2 × KY-040 rotary switch **magpi.cc/ky040**
- MCP602 **magpi.cc/mcp602**
- MCP4921 DAC **magpi.cc/mcp4921**

Make Pico-Voice, your own voice altering machine to add filtering, reverberation, echoes, pitch changing, and even sound like a Dalek. Oh, and did we say, it can talk backwards in almost real time?

01 Digital signal processing

So far we have looked at programming Raspberry Pi Pico with MicroPython, but when you step up to using C, the system is magnitudes faster. This allows you to do some DSP (digital signal processing) on a live audio input to produce all sorts of effects. As Pico's RP2040 has a total of 256kB of on-chip SRAM, it has sufficient memory so that a lot of effects can be made without resorting to external memory. We can use the on-board ADC (analogue-to-digital converter) to get our sounds in for processing, and an external DAC chip to turn the digits back to sound.

02 Block diagram

Figure 1 shows the basic block diagram of what we are going to build. While much of the work is done by our Pico, we need to be able to choose what effect we want, alter some basic parameters for the effect, and see what we have chosen. We chose an external DAC chip connected to the SPI bus, using the I2S interface, on the grounds of cost and ease of use. While the chip we used has only 12 bits of resolution, given the system noise and the fact we are using only voice, it means you don't pay the price in quality.

03 Programming in C

We also added a reset switch to remove the requirement of keep disconnecting the USB connector to put Pico into boot mode. However, programming Pico in C is not as easy as it could be. If you are use to Python's 'Run' command, or the Arduino's click for compile and run, be prepared for a lot more work. Along with having to bother about the code, there are the almost incomprehensible make files to create, so that the compiler knows where to find the library files. As there are plenty of online tutorials about this, we will not duplicate it here.

04 The circuit

The circuit diagram of the Pico-Voice is shown in **Figure 2**, and there is a lot going on here. We used the MCP602 chip for both the microphone amplifier and the headphone amplifier, and the

> "The rotary encoders and BCD switch required a bit of hardware debouncing"

MCP4921 DAC. The rotary encoders and the BCD (binary-coded decimal) switch all required a bit of hardware debouncing. The coupling and filter capacitors in the amplifier should be of film type like Mylar or polypropylene, but all the others can be ceramic. A mid voltage bias point is made from two 1 kΩ resistors, while a 33 kΩ resistor controls the gain of the microphone.

05 Construction design

The idea is that the construction is a single board without any flying leads, so that all the

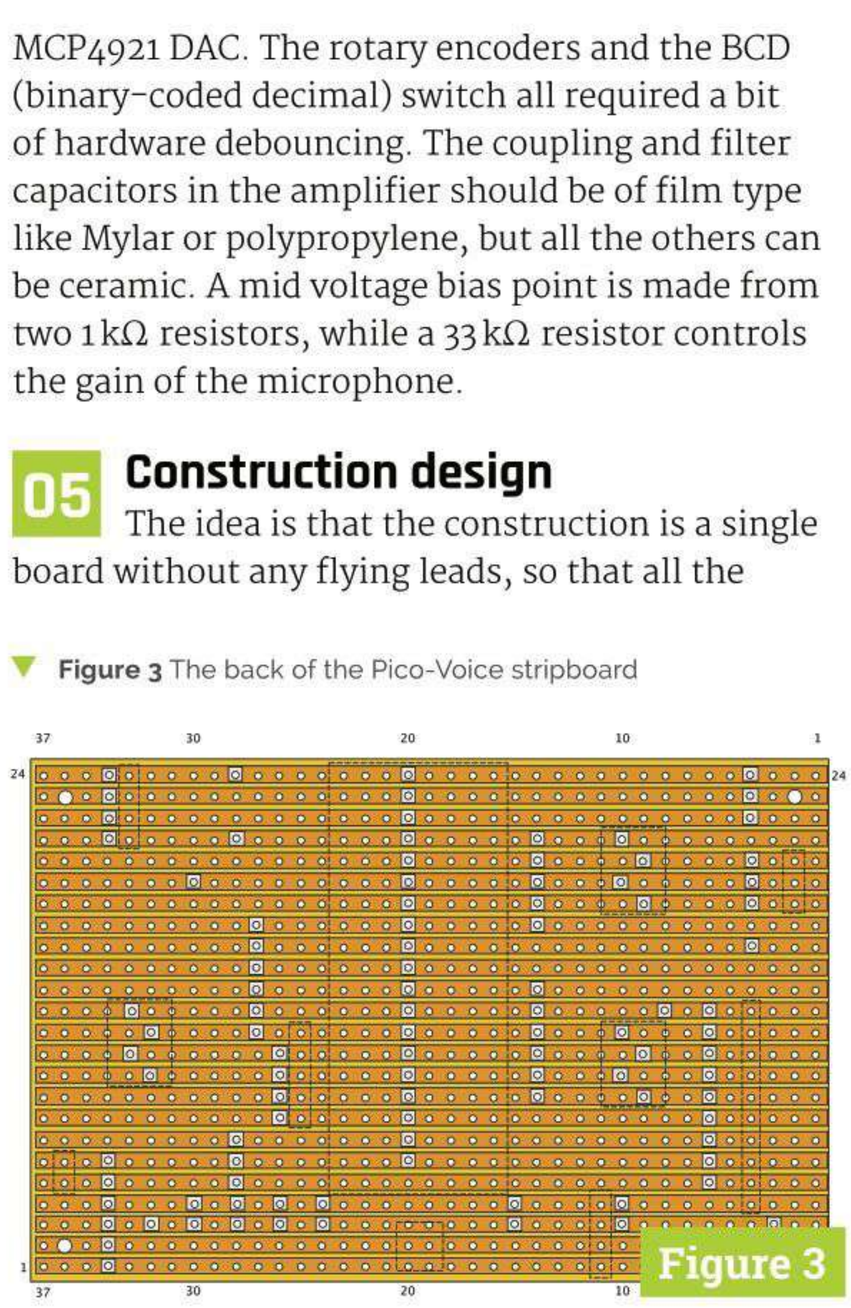


Figure 3

▼ **Figure 3** The back of the Pico-Voice stripboard

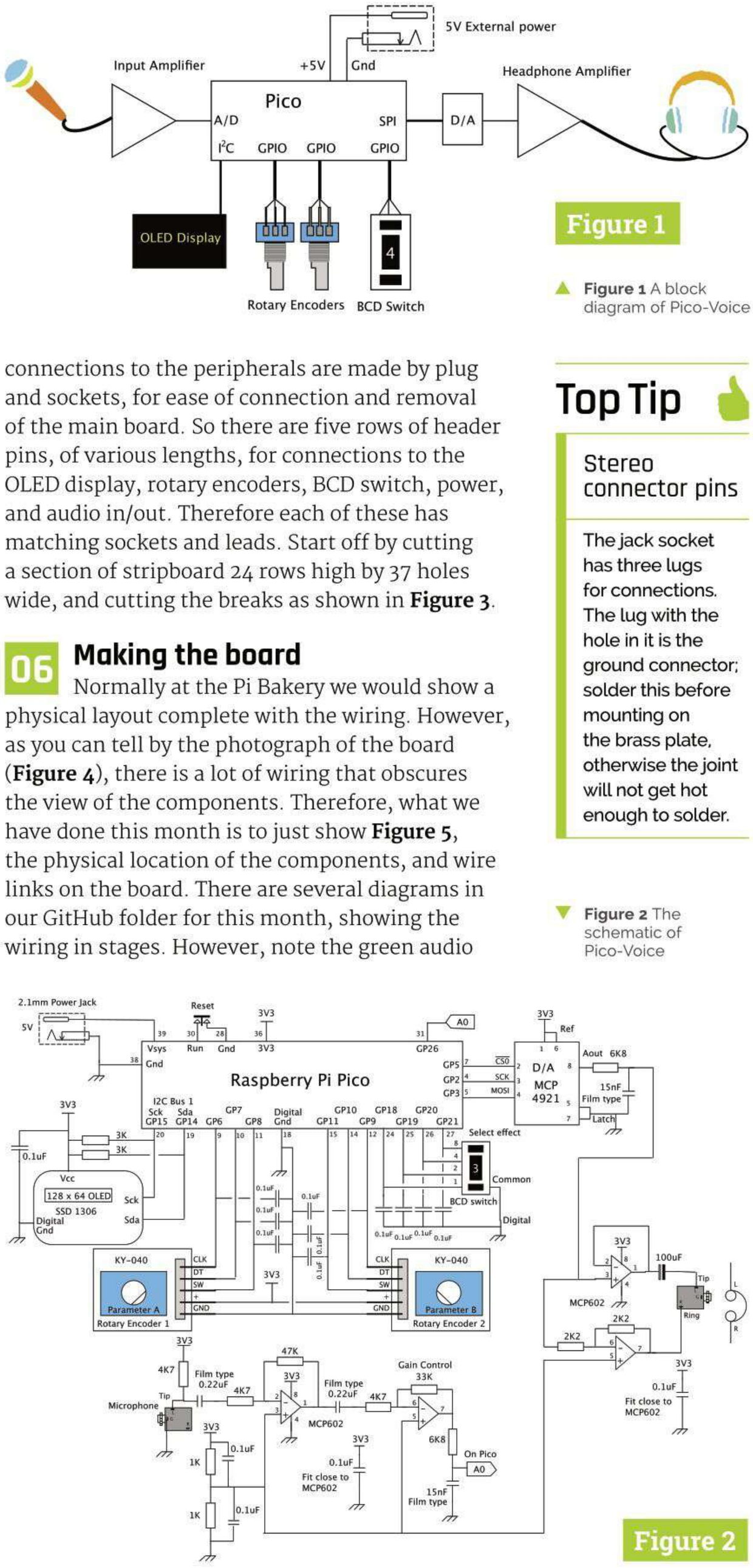


Figure 2

▼ **Figure 2** The schematic of Pico-Voice

connections to the peripherals are made by plug and sockets, for ease of connection and removal of the main board. So there are five rows of header pins, of various lengths, for connections to the OLED display, rotary encoders, BCD switch, power, and audio in/out. Therefore each of these has matching sockets and leads. Start off by cutting a section of stripboard 24 rows high by 37 holes wide, and cutting the breaks as shown in **Figure 3**.

06 Making the board

Normally at the Pi Bakery we would show a physical layout complete with the wiring. However, as you can tell by the photograph of the board (**Figure 4**), there is a lot of wiring that obscures the view of the components. Therefore, what we have done this month is to just show **Figure 5**, the physical location of the components, and wire links on the board. There are several diagrams in our GitHub folder for this month, showing the wiring in stages. However, note the green audio

Top Tip

Stereo connector pins

The jack socket has three lugs for connections. The lug with the hole in it is the ground connector; solder this before mounting on the brass plate, otherwise the joint will not get hot enough to solder.

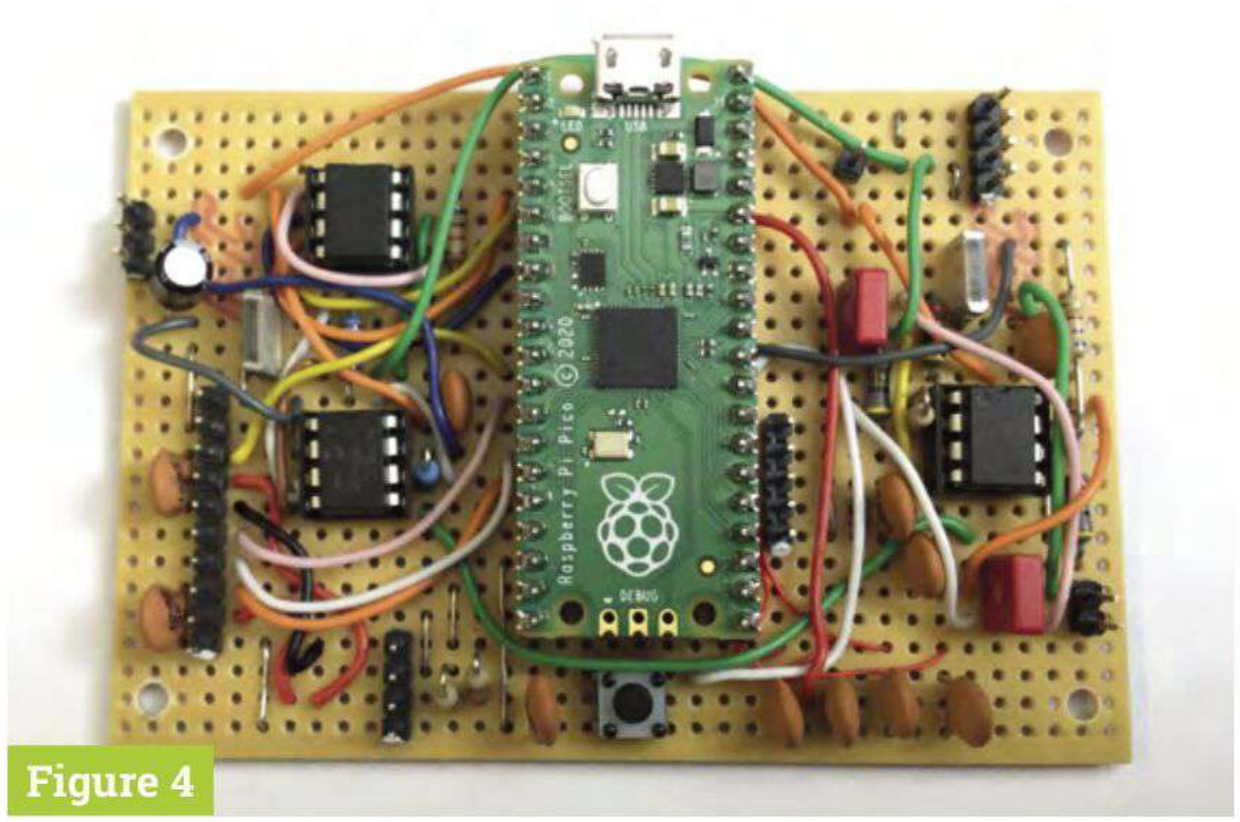


Figure 4

output wire, as it is used in an unusual way, with the wire bridging the gap between tracks.

07 The enclosure

The plans for the box are on our GitHub page (**magpi.cc/pibakery**). While a functioning box can be made by cutting out 3 mm plywood with a fretsaw, to make the badge – a homage to similar decorations found on VOX and Marshall amplifiers – you will most probably need to have access to a laser cutter. One problem we faced was that we couldn't find 3 mm jack sockets that could be mounted through a 3 mm thick panel. Therefore, we had to make a plate from 1 mm brass to mount the socket. A 6 mm hole and two 2.5 mm holes fixed the plate to the side of the box. See **Figure 6**.

▲ **Figure 4** Photograph of the front of Pico-Voice

▼ **Figure 5** Diagram of the fixed components of Pico-Voice

08 Board peripherals

To save space on the board, we mounted two of the debounce capacitors on each rotary encoder between ground and the signal pins CLK and DT (**Figure 7**). If you look closely at the BCD switch, you will see two plastic bumps where you can stack other switches to cover more digits; these need to be cut off close to the switch body with side cutters, to make sure the body of the switch goes through the hole in the top panel. You also need to cut a recess for the OLED display so it fits flush on the panel.

09 Box assembly

We painted the box sides before gluing them together in a ring, using the box top and bottom to ensure the sides were vertical. The top and bottom are held together with 30 mm hexagonal spaces and are not glued to the sides. We painted the badge gold and black, and glued it down to the top lid.

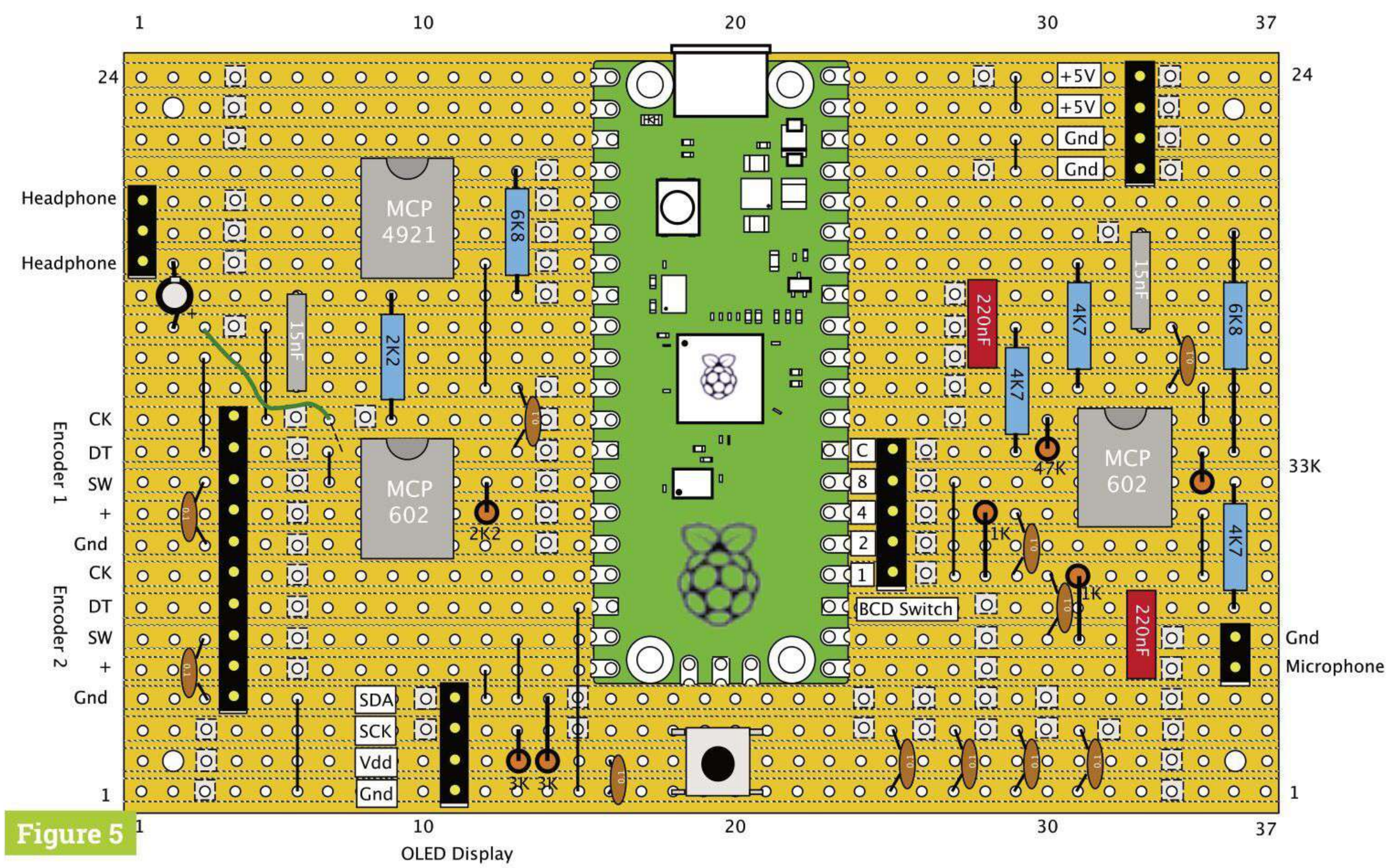


Figure 5

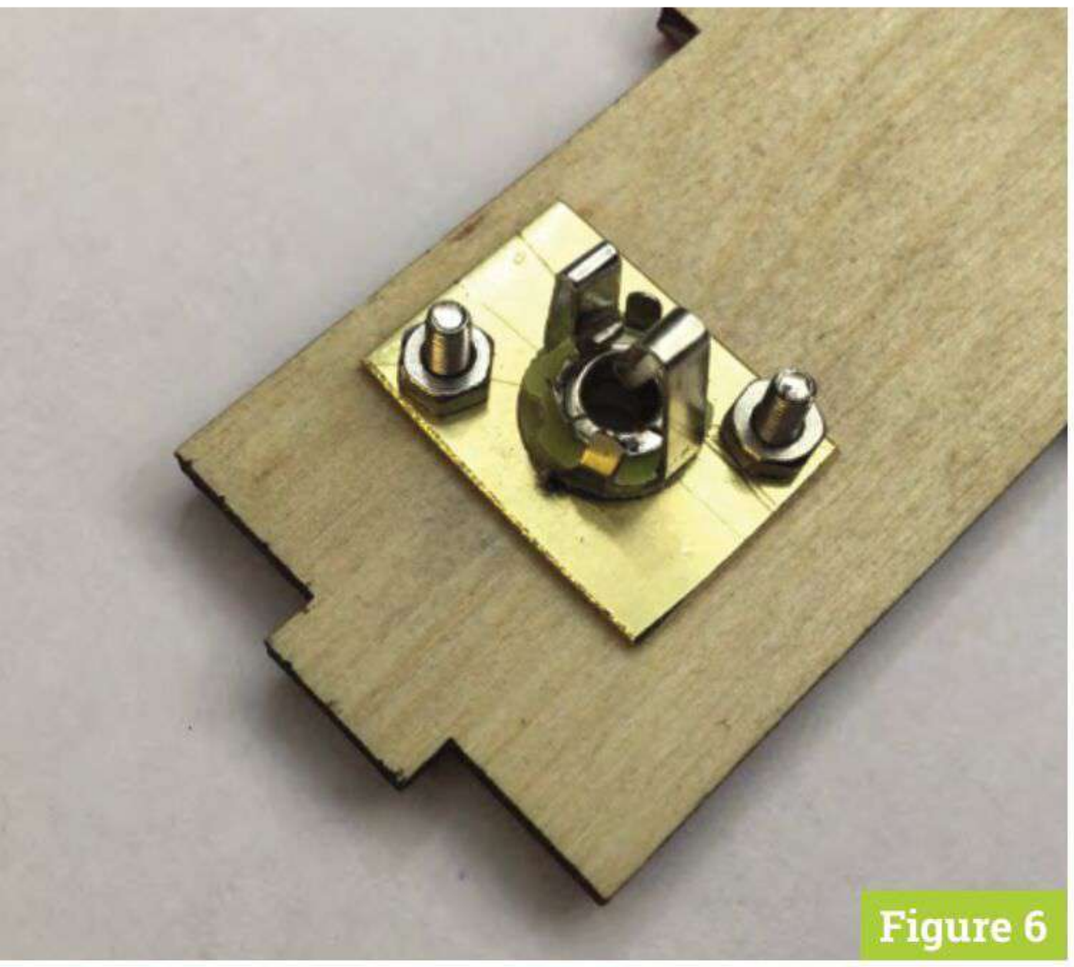


Figure 6

Then the circuit board was placed so that the USB connector was central to the hole in the back. We then drilled through the mounting holes into the base, and mounted it on 6mm plastic spacers. Hot-melt glue holds the OLED and BCD switch in place. See **Figure 8**.

10 Variations

You can make some variations to the Pico-Voice, to customise it for your own purposes. For example, you could include a 3×AAA battery pack to power it, like we did when making the MIDI Touch Guitar. You can get some improvement in noise performance by powering the analogue circuit with its own 3V3 regulator. The wiring has been laid out so that this will involve just a single wire change. Finally, the analogue section could be built on a separate board. We used a headphone/microphone headset, but you can use combinations of separate microphones and powered computer speakers.

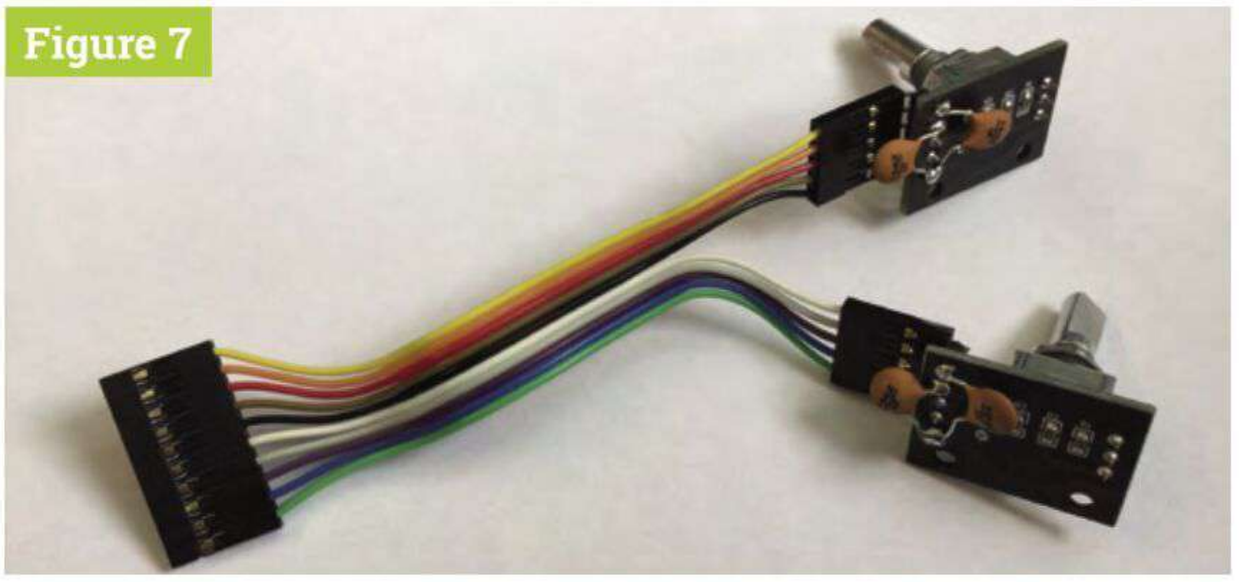


Figure 7

◀ **Figure 6** Mounting the audio jack socket

▲ **Figure 7** Rotary encoders showing debounce capacitors to ground on CLK and DT pins of Pico-Voice

11 Next Month

Next month we will look at the software techniques needed to read the encoder and switches. While we covered how to read rotary encoder switches in *The MagPi* #92 (**magpi.cc/92**), this was using Python on a Raspberry Pi computer and the 'pigpio' library. This is not available for Pico – and besides, it is much more capable than we need. Finally, we will see how the various effects and variations work. These include an octave shift up or down to change the pitch of your voice, how to make delays and reverberation effects, and wacky distortions. See you then.

Warning!
Power tools

Be careful while using power tools during this build.

magpi.cc/drillsafety

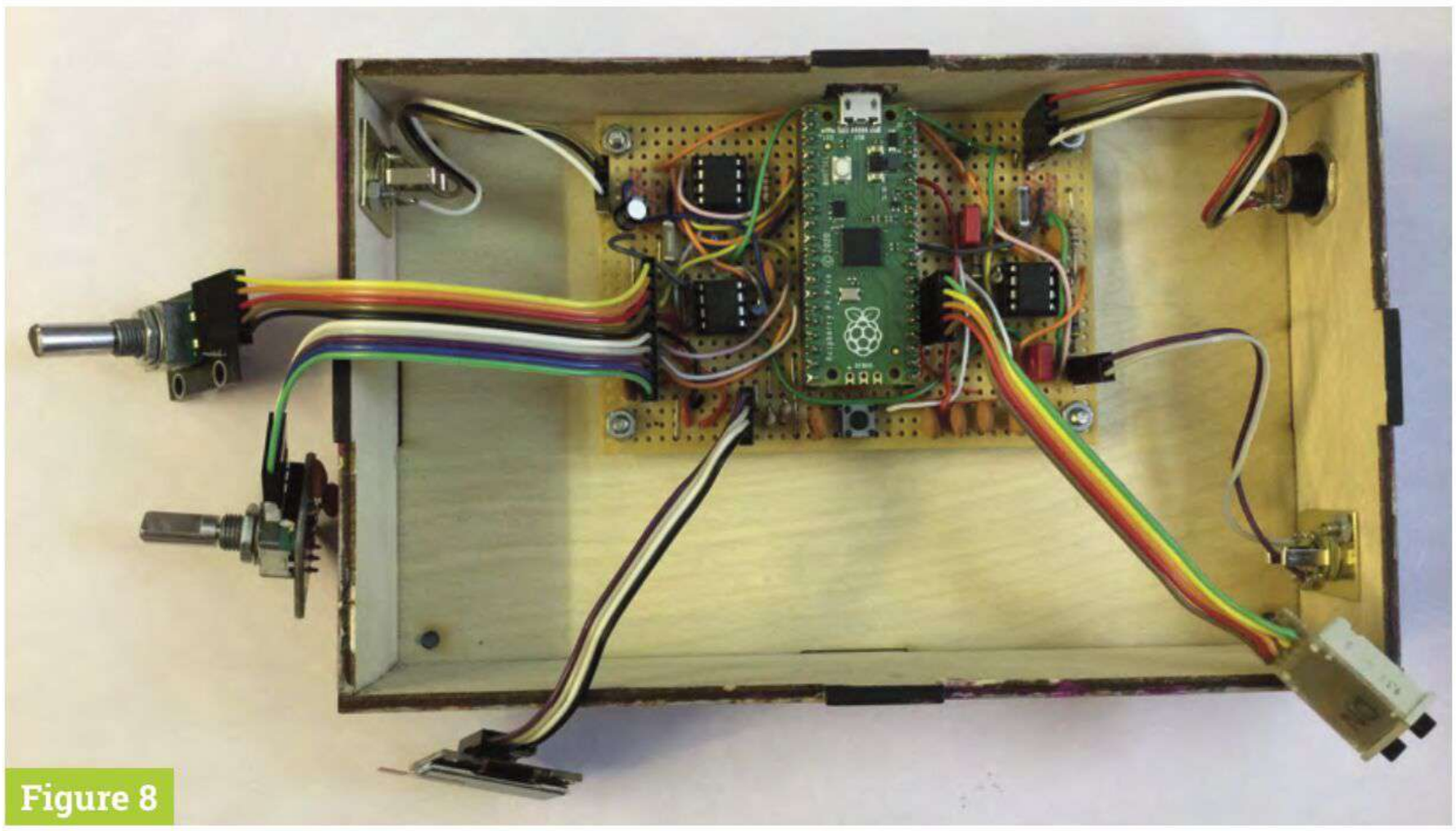


Figure 8

◀ **Figure 8** Circuit board and all peripherals mounted on ribbon cable connectors

Summer Projects

With summer just around the corner, our minds have turned to getting out and about and enjoying the better weather – and which Raspberry Pi projects can help

By **Nik Rawlinson**

Summer is fast approaching – and that's the perfect excuse to get building. Whether you want to spy on your local wildlife, upgrade your vegetable patch, or feed your fish when you're off on a weekend break, Raspberry Pi and a handful of add-ons make a great starting point. We've packed the next eight pages with some of the most inspirational projects to be found. They include a smart tide monitor, which will tell you when's the best time to hit the beach, and a clever Heater Meter that can keep an eye on your barbecue while you get on with the prep in the kitchen.

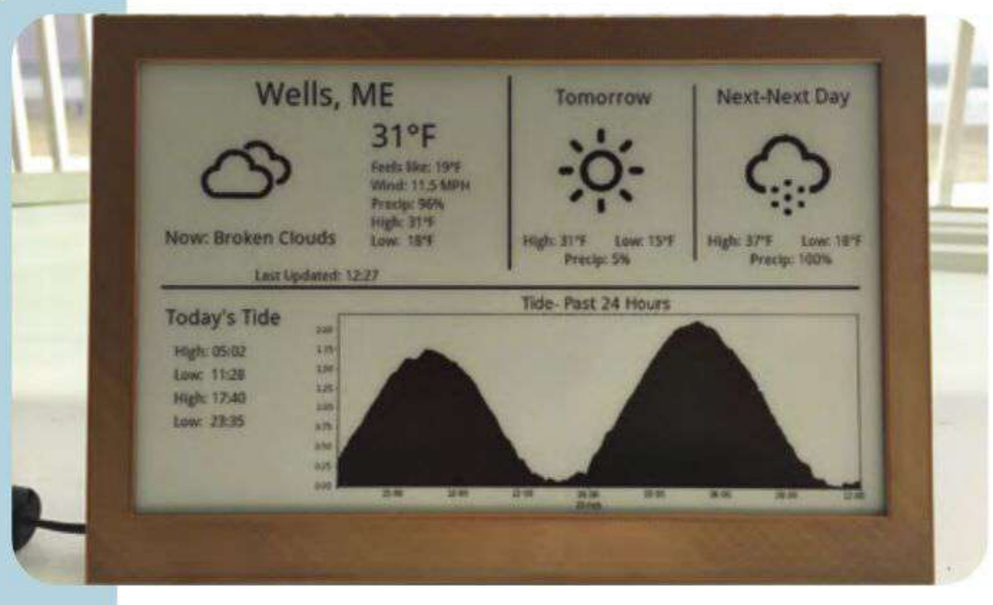
Makes for when..
you're on the move

Use Raspberry Pi to plan the best time to hit the beach, or give your preferred choice of transport some serious smarts

Check the tides

If you're heading out for any kind of water-based activity (and that includes sitting on the beach), it helps to know whether the tide is in or out and which way it's heading. Sam Baker's neat e-ink tide and weather tracker (**magpi.cc/tideweather**) uses Raspberry Pi Zero and an enormous (7.5 in) e-ink display to track the motion of the ocean and upcoming weather conditions, so you don't arrive at the beach to find the sand submerged.

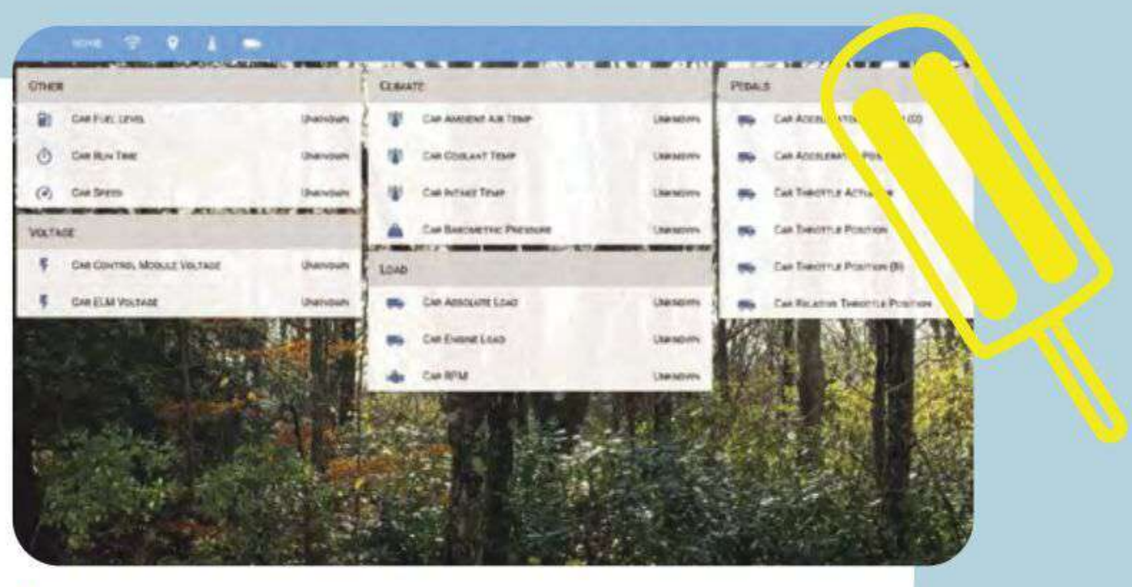
▼ Avoid nasty surprises when you arrive at the beach: check the tide level before you leave home



Build a smart boat...

OpenPlotter (**magpi.cc/openplotter**) describes itself as an open-source sailing platform for ARM computers, of which there are several builds optimised for different versions of Raspberry Pi. If you're getting your boat back on the water after months in storage, this would be the ideal time to consider an upgrade.

It brings together a wide range of functions critical to owning and operating a boat, like plotting a course, downloading weather monitoring instruments through a unified dashboard, accessing SDR (software-defined radio), and even sending boat data to your phone using Telegram or email when you're not on board.



▲ Van Home Assistant can monitor a range of metrics from right around the van to present its status at a glance

...or a smart camper

Not all of us are lucky enough to own our own boat, but if you have a camper van, or plans to convert a second-hand van into the perfect post-lockdown staycationer, Van Home Assistant (**magpi.cc/vanassistant**), the code for which is open-source and available for download on GitHub, would make a great addition. Running on Raspberry Pi, it detects motion to act as a burglar deterrent, interfaces with an OBD (on-board diagnostics) dongle to read engine and vehicle diagnostics, monitors internal temperatures (to make sure it's a safe environment if a dog is left in there alone), and keeps an eye on how much cellular data it's using to share all this information.

▼ Raspberry Pi can sit at the heart of a smart boat, adding essential navigation, communication, and control features

"Track the motion of the ocean and weather conditions"

"Use Raspberry Pi to make sure pets are properly fed and watered"

Pet control

Summer is the time for holidays – even if that might mean holidaying closer to home this year. If you have the kind of pet that won't pine for you when you're away, like fish or rodents, you can use Raspberry Pi to make sure they're properly fed and watered.

Explaining Computers has put together a comprehensive video (**magpi.cc/pizerohamster**) that shows you how you can keep your hamster fed while you're away by controlling the doors on a pair or boxes containing food using Raspberry Pi Zero and a couple of servo motors. Holding back food until specified feeding times prevents your furry friend from gorging itself the first day you're gone, and spending the rest of your break hungry.

For larger pets, like cats and dogs, check out Rob Peck's Petfeedd. The code that makes it work is on GitHub (**magpi.cc/petfedd**) and the fully illustrated instructions, which involve drilling, gluing, and painting, are on Rob's blog (**magpi.cc/petfeeders**).

If you have an aquarium, reef-pi (**magpi.cc/reefpi**) uses a Raspberry Pi Zero to monitor temperature, light levels, and water chemistry to make sure more demanding occupants, like corals, have the ideal conditions for a long and healthy life. If you prefer not to build your own, you can alternatively buy plug-and-play hardware for running reef-pi at home (**magpi.cc/reefpihardware**).

▲ Rob Peck's Petfeedd is a homemade build for cats and dogs

▶ Plug-and-play hardware like the Robo-Tank Aquarium Controller simplifies the process of implementing reef-pi

In the garden

With some smart housing, a few cables, and some sensors, your Raspberry Pi can be an essential garden assistant

Print your own lawn-mower

The return of summer means an addition to your weekly to-do list: mowing the lawn. But not if you build a PiMowBot smart lawn-mower robot (**magpi.cc/pimowbot**). This uses any Raspberry Pi to control an autonomous lawnmower that navigates your garden using GPS and offers optional remote control, so you can keep the lawn trimmed from the comfort of a garden chair. The hardware, comprising the chassis, cutterbar and so on, is solar-powered and can be 3D-printed, while the software is a €19.99 download. The OBJ (Wavefront OBJect) file patterns for the various parts you'll need to print are a £29.46 download from Cults3d (**magpi.cc/pimowbotcase**). You'll need to buy several components to put it together, as it also relies on a number of sensors, including – aside from the GPS receiver – a temperature and humidity sensor, compass module, and Camera Module.

Although you do need to invest in quite a few parts for the PiMowBot, and spend time assembling them, the project still manages to undercut (sorry!) commercial alternatives, for which prices start at around £500, by a considerable margin.

▼ Sit back, put your feet up, and enjoy the sunshine while PiMowBot takes care of mowing your lawn

▲ Never forget to water your plants again. Pimoroni's Grow Kit uses hypersensitive sensors to detect moisture levels in your pots and display the output using an easy-to-glance-at bar graph

...and look after your plants, too

Pimoroni's Grow Kit (£30, **magpi.cc/grow**), which is also available in board-only form (£19.50) or with seeds for growing herbs or chillies, is a great first project if you've never built with Raspberry Pi before. Pairing three capacitive moisture sensors, its integrated screen shows you at a glance whether your pots need attention. The Grow Kit PCB is the perfect size for pairing with Raspberry Pi Zero, and it has both a light sensor and a piezo buzzer, the latter of which will chirp when it's time to give your plants a splash of water.

If you're feeling particularly adventurous, you can go one step further. Flip over the PCB and you'll find three connectors with which you can control three 5 V electronic devices. Why? Because with some nifty programming you could activate them when the detected moisture level falls below a set threshold, to turn on a pump, or a grow light when the light sensor detects a cloudy day.

This could be the perfect solution, both for wannabe windowsill gardens who lack the required green thumbs, and anyone heading off on holiday who doesn't have neighbours to pop in and look after their plants.

Become self-sufficient

If you're even more ambitious, and have dreams of living *The Good Life*, MudPi (**magpi.cc/mudpi**) will make your life a whole lot easier, by connecting Raspberry Pi to a series of sensors courtesy of a dedicated board (or boards). Once set up, you can leave your crops while you have a few days' break, safe in the knowledge that the all-important job of keeping them watered will be taken care of. But it's not just an automated watering system: MudPi turns your plot into a full-blown smart garden by gathering all manner of metrics with which you can monitor growing conditions and make better informed gardening decisions, which can help you save money by reducing waste.

The system is complete, while still being actively developed, with a web-based interface called Solarbeam currently in the works. Once ready for roll-out, this will let you do a lot of minor jobs from indoors (not weeding, sadly). Best of all, MudPi is designed to integrate with existing irrigation systems, so if you already have valves and piping in place, there's a good chance you can upgrade, rather than replace them.

If you live in a flat with a balcony and don't have access to a dedicated vegetable garden, something smaller may be called for. In that case, check out the automated Raspberry Pi Greenhouse project (**magpi.cc/automaticgreenhouse**), which is suitable for spaces as small as half a metre square.

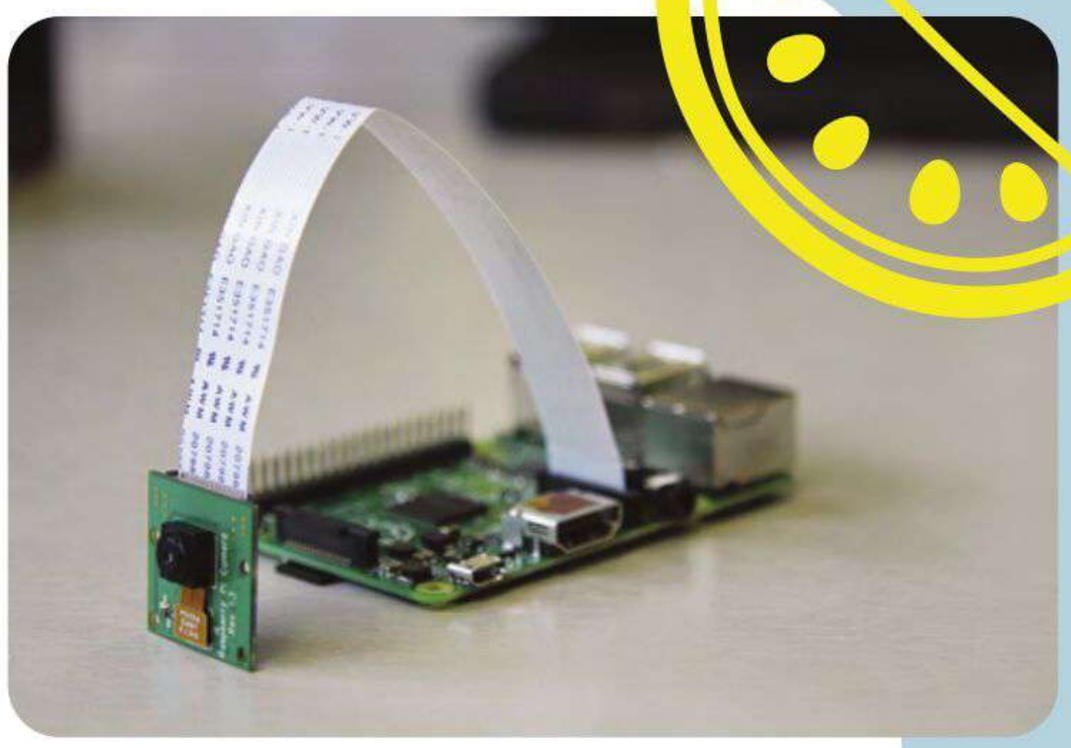
▼ If you have dreams of self-sufficiency, MudPi can help, by monitoring your garden and automating common tasks

Take pictures

Set up your Raspberry Pi to detect movement and take pictures automatically, and you'll never miss a winning shot again

Build a trail camera

One of the best things about summer is the return of a host of migratory birds that desert us in the colder months. And, while foxes and badgers will have been with us throughout the winter, hedgehogs will have been hibernating between late autumn and early spring. Many of these animals are timid, so spotting them requires that you get up early, stay up late, or set up a trail camera which uses motion detection to capture an image when they pass. This has been a popular use for Raspberry Pi for years, but there are so many ways to go about it, you might be wondering which are the best options. You can pick up all the parts you need to build your own trail camera – aside from the power supply – from The Pi Hut for £110 (**magpi.cc/naturebytes**), or the case on its own for £40 (**magpi.cc/naturebytescase**) if you have most of the other required components knocking around from old projects.



▲ Raspberry Pi Camera Module is ideal for tracking wildlife. Add power and place it inside a plastic food container to keep it safe

▼ The Naturebytes camera case keeps all the components of an automated bird and wildlife camera neat and tidy

Put your bird table on Twitter

Choosing the best position for your camera is an art in itself. If you live near some secluded woodland and have permission to site a camera there, strapping it to a tree a couple of feet off ground level will give you a good chance of capturing deer and other four-footed wildlife. For birds, try mounting the camera on or beside a feeding table or, as this Camera Trap project recommends (**magpi.cc/cameratrap**), use a plastic food container. Depending on the size of the container, this may also allow you to house a high-capacity battery pack for several days' use, and an SSD drive if you need more capacity than your SD card allows. This solution – and an alternative from Peak Nature (**magpi.cc/trailcamera**) – is ideal for positioning on the ground, where it will capture both larger birds, like pigeons, and hedgehogs.

We were particularly taken by Nick Vasilyev's social media-connected bird table project (**magpi.cc/birddetector**), which uses a Raspberry Pi Camera Module as a motion detection system to constantly keep a digital eye on his bird table. When it spots the arrival of a feathered friend, it grabs a few seconds of video and posts them to Twitter. That's what we call tweeting.

> **"Mount the bird-box now and leave it in place for a year"**

▲ Give your bird-box the *Big Brother* treatment by fitting it with an infrared camera, and you can spy on your feathered friends

Recycle leftovers into a wildlife camera

Available from the Pimoroni website, the My Naturewatch Wildlife Camera kit (£34.50, **magpi.cc/naturewatch**) includes everything you need to build a tiny camera, including the Raspberry Pi Zero W that will host it. The result can be housed in a smaller food container, with the cut-off neck of a plastic bottle as a lens protector, or a jam jar, where it should be impervious to the elements. All you need to provide is a small USB power bank.

None of the components are particularly exotic and you may well have them all already, either lying around following retirement from previous projects you've lost interest in, or surplus to requirements. If that's the case, then head over to My Naturewatch, where you'll find the instructions for making the project detailed in full (**magpi.cc/mynaturewatch**).

▶ If you want to build your own wildlife camera, check out the instructions from My Naturewatch (image: mynaturewatch.net)

Build an infrared bird-box

You can pick up a wooden bird-box at hardware stores for well under £10 (**magpi.cc/birdbox**) and, with a Raspberry Pi NoIR Camera Module, give it a serious upgrade with Marc Scott's Infrared Bird Box project on GitHub (**magpi.cc/irbirdbox**). As the NoIR camera works well in low light, you may be lucky enough to be able to watch a couple of parent birds lay and hatch a clutch of eggs – but you'll have to be quick. If the birds have already been and gone in your area, don't be put off: often, animals can be spooked by new additions to the garden, so mounting the box now and leaving it in place for a year in the run-up to next year's nesting season may increase your chances of seeing something in early 2022. Don't be tempted to upgrade an existing box if it's currently in use.

Be a solar squirrel spy

Unless you're able to site your camera near a power supply (on the side of your shed, for example), you'll need to come up with an alternative power source. A high-capacity USB power bank is ideal or, if you've got a sunny garden, you might like to go solar. Mike Sadowski, in *The MagPi* issue 103 (**magpi.cc/103**), has created a project for a ML-based Bird and Squirrel Detector (**magpi.cc/squirrelkam**) using Raspberry Pi and Amazon Web Services. Amazon Rekognition replies with a list of 'labels', and the code looks at the labels and decides if the image contains a bird or squirrel.

◀ Keep an eye on your local squirrel population with this bird and squirrel detector

Live the outdoor life

Take advantage of the better weather by equipping your Raspberry Pi with solar, or using it to help you cook the perfect BBQ

▲ Run your Raspberry Pi indefinitely, so long as the sun keeps shining, by connecting a battery and solar panel

Add solar power to your Raspberry Pi

Snapping squirrels isn't the only reason to add solar power to your Raspberry Pi. In fact, it's the perfect solution to hosting projects away from a mains power source, like internet-connected thermometers that will warn you when your greenhouse is getting too hot. The parts are easily sourced, and building the kit should take less than half an hour. You can find a full set of instructions – and a shopping list – at Howchoo (**magpi.cc/solarpower**). Also take a look at Kaspars Dambis's Solar Powered Camera project on *The MagPi* website (**magpi.cc/solarcam**).

BBQ safely

One thing that's certain to put a dampener on a summer get-together is barbecued food that's charred on the outside and raw in the middle. Fortunately, a lot of makers have set themselves the task of solving this problem – which they've done with aplomb. Tempiture (**magpi.cc/tempiture**) pairs Raspberry Pi with a breadboard, food probe, and a handful of resistors to produce a wireless grilling thermometer which sends readings to the web. As a barbecue can take hours to get to cooking temperature, this lets you keep an eye on its progress while you're prepping food in the kitchen.

It's not your only option, either. PitmasterPi (**magpi.cc/pitmasterpi**) performs a very similar job, taking regular readings to populate a real-time dashboard, and optionally sending emails or texts at crucial moments.

HeaterMeter (**magpi.cc/heatermeter**) pairs Raspberry Pi with an Adruino microcontroller, thermal probe, and fan to maintain perfect temperatures, with support for web streaming, graphing, and alerts. What's particularly appealing about HeaterMeter is that you can choose different starting points for your project, depending on how confident you are. If you're a dab hand at soldering and reading a circuit diagram, start from scratch with a kit; but if you're just craving a burger, skip all that and opt for a fully assembled board instead.

▼ HeaterMeter lets you keep an eye on your BBQ from a distance, freeing you to get on with prep in the kitchen while the HeaterMeter maintains cooking temperatures

THine Cable Extension Kit

SPECS

BOARDS:
Transmitter Board with THine THCV241A MIPI CSI-2 to V-by-One HS Serializer; Receiver Board with THine THCV242 V-by-One HS to MIPI CSI-2 Deserializer

COMPATIBILITY:
Camera Module V1.3 (limited support); Camera Module 2.1; HQ Camera

DIMENSIONS:
TX board: 38×25 mm; RX board: 65×56 mm

▶ THine ▶ **magpi.cc/cableextender** ▶ £51 / $59

Connect an HQ Camera to Raspberry Pi via an extremely long Ethernet cable. **Lucy Hattersley** takes a look at this long shooter solution

THine Cable Extension is a kit for Raspberry Pi that swaps out the CSI cable (Camera Serial Interface) for a LAN/Ethernet cable. This enables you to dramatically increase the length of the cable that can be used, positioning a Camera Module up to 20 metres away from Raspberry Pi.

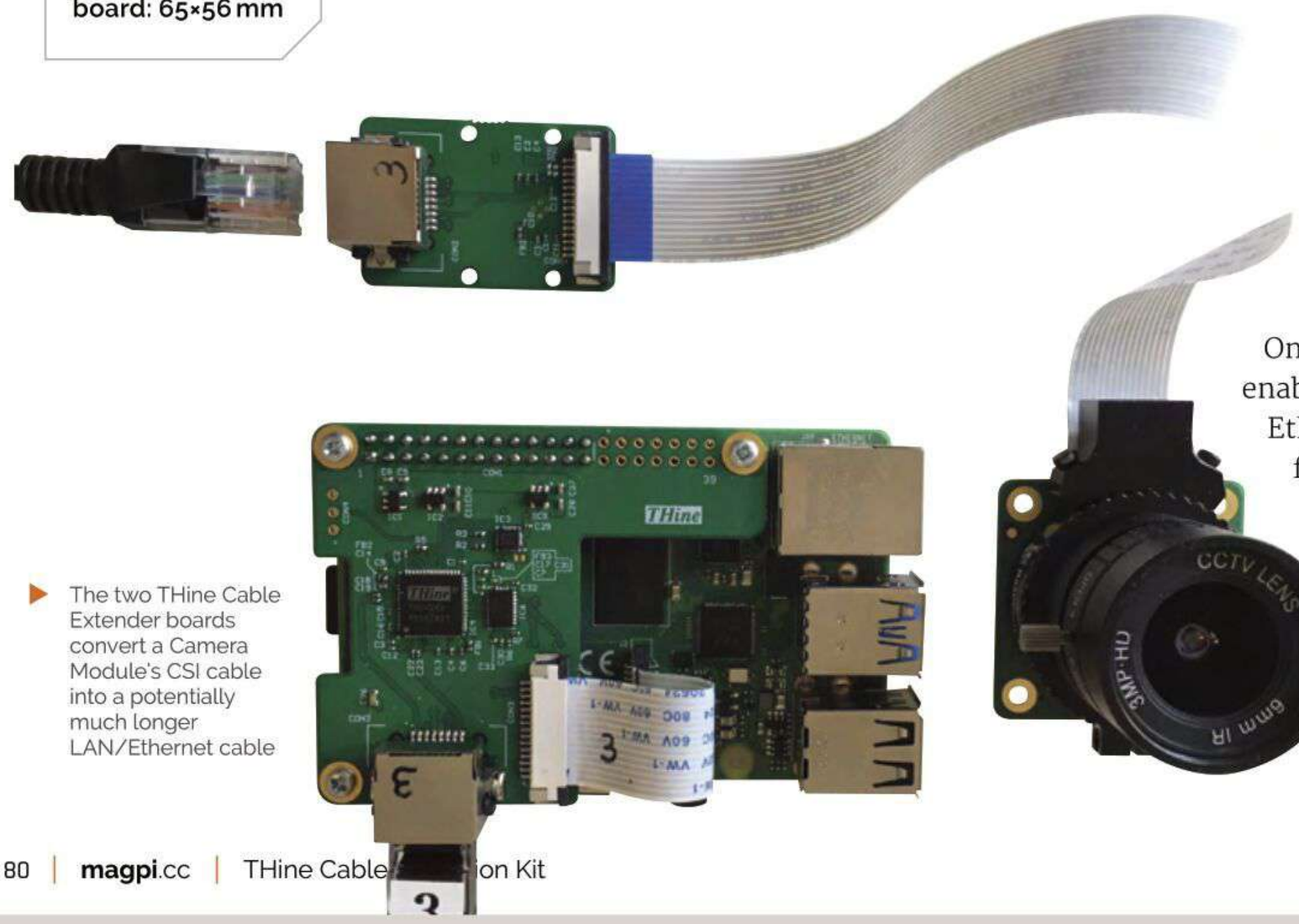
The kit comes in two parts: the Transmitter Board attaches to your Raspberry Pi Camera Module / HQ Camera; the Receiver Board connects to Raspberry Pi via the GPIO pin header.

Each board features a CSI socket and RJ45 (Ethernet) socket and the two are connected via a CAT5e or higher LAN/Ethernet cable. The system is designed to be 'plug and play', so all you need to do is hook everything up and use the Camera Module as normal.

Japanese company THine's V-by-One HS Serializer / Deserializer technology provides a "high speed data link solution for MIPI CSI-2, camera serial interface". Learn more on THine's website: **magpi.cc/THCV241A**.

The Cable Extension kit comes supplied with the two boards, a 2-metre LAN cable and the ribbon flex cables, along with mounting screws. Our test kit also came with a Raspberry Pi 4, HQ Camera, and a 5 m Elecom CAT6 Ethernet cable.

We took everything apart and set it back up with a fresh installation of Raspberry Pi OS, then followed the quick-start guide (**magpi.cc/cableextensionqs**).



▶ The two THine Cable Extender boards convert a Camera Module's CSI cable into a potentially much longer LAN/Ethernet cable

Snap happy

Setup was hassle free and it 'plugged and played' as outlined by THine. There was no need for any additional software installation.

One thing of note: this is not a network-enabled solution. Instead, the LAN/Ethernet cable is designed to run directly from Raspberry Pi to Camera Module.

We tested it around the house and ran the cable out into our garden where we kept an eye on a bird feeder from afar.

And 20 metres is a massive upgrade from the 20 cm CSI cable included with the Camera

THine Cable Extender is ideal for running a Raspberry Pi HQ Camera outdoors while Raspberry Pi remains safely out of the elements

> Works exactly as outlined and performs a useful function admirably well

Module. We checked with Raspberry Pi, and there is a limit with CSI before you lose signal integrity. There's no guarantee that a CSI cable longer than the 20 cm one supplied will work.

So this is a big upgrade in effective distance. The Cable Extension Datasheet (**magpi.cc/cableextensiondatasheet**) has more info on cables tested up to 20 m with this kit.

It can be a mild trouble to fine-tune the HQ Camera Module's adjustment rings with the Raspberry Pi screen so far away. Apart from that, we struggle to find any downside. Cable Extension Kit works exactly as outlined and performs a useful function admirably well.

There are many use cases where it is beneficial to keep the camera and Raspberry Pi some distance apart, especially when filming in a hostile environment. THine outlines a project where they set up an outdoor PiKon 3D-printed telescope (**pikonic.com**) and used it at night during a freezing Chicago winter: **magpi.cc/cableextensionpikon**).

THine Cable Extension Kit is a niche product for a niche use case. But the technology is clever and it works exactly as outlined with no fuss. If you have a desire to place your Camera Module some distance away from Raspberry Pi, then this is the way to do it.

Verdict

A breeze to set up, and works exactly as outlined by THine. The V-by-One HS technology is interesting and this is the way to go about connecting a Raspberry Pi Camera Module over a long distance.

9/10

10 Amazing: Media player projects

The best ways to groove, dance, chill, and more using a Raspberry Pi

You can use Raspberry Pi to play all kinds of media thanks to its small but mighty system-on-a-chip. We've seen people use Raspberry Pi to create incredible and inventive projects involving media – here are some of the very best.

▼ Big and Smart Raspberry Pi Picture Frame

Digital photo supremacy

With a spare monitor, a nice frame, and a Raspberry Pi loaded with photos or connected to a cloud service, you can easily have a nice, ever-changing series of pictures displayed on your wall.

magpi.cc/bigframe

▲ Lunchbox Arcade Game

Lunch-time fun

There are many great and cool video game projects, but this portable arcade machine disguised as a lunchbox is still one of our favourites. Playing it at home or at work on your break seems like the perfect fit for it.

magpi.cc/lunchbox

▶ iPod Classic Spotify music player

Upcycled classic

The clicky wheel iPod is a very important device in the history of portable media players. So of course, Guy here ripped one apart and upgraded it with a Raspberry Pi Zero W and Spotify.

magpi.cc/ipodspotify

▲ BOSEBerry Pi

A 2010 throwback

The iPod docks of yore are remembered fondly by many and are still available in a lot of hotel rooms. The form factor (and used price) really appealed to maker David Hunt, who turn one into an internet radio box.

magpi.cc/boseberrypi

Atomic TV

Custom-built nostalgia

Atompunk is a style that takes inspiration from American ideas of the future from the 1940s and 1950s. This retro-future aesthetic has been perfectly captured in Atomic TV, with bezels and curves and switches to match. It's made from scratch as well and wouldn't look out of place in the Carousel of Progress.

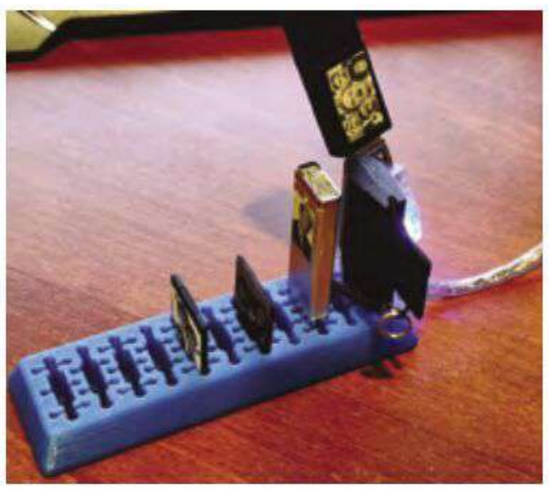
magpi.cc/atomictv

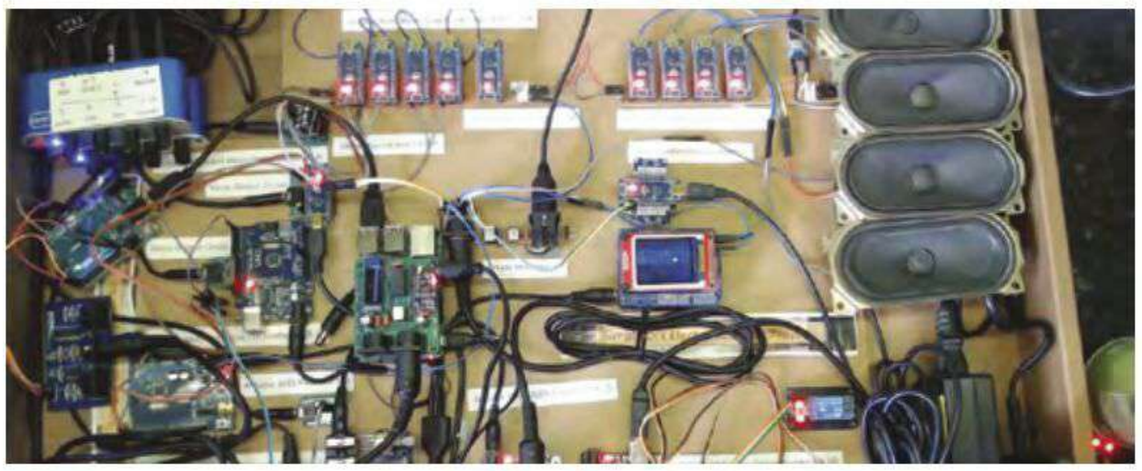
PiVidBox

Simple solution

This basic media player just requires you to plug in an SD card or USB stick with various media on it. It uses old storage that would otherwise never get used, and each has a curated selection of content, labelled to make sure you know what's on it.

magpi.cc/pividbox





Lo-Fi Orchestra

Impractical yet cool

Using a whole host of microcontrollers and microcomputers – including Raspberry Pi – Kevin, aka DIY Electro Music, has managed to recreate a lo-fi version of Gustav Holst's *The Planets* suite. He's making his way through the suite, but *Jupiter* is one of our favourites.

magpi.cc/lofiorchestra

Vintage television

Digital to analogue

Portable TVs were the coolest thing in the eighties and nineties. However, for obvious smartphone-related reasons, they have fallen out of favour. This 1975 portable TV has been upcycled by the great Martin Mander to use a Raspberry Pi and a TV HAT for portable digital telly. Nostalgic.

magpi.cc/vintagetv

Vinyl emulator

RFID records

Using Sonos and Spotify and some very lovely, RFID-enabled cards to represent records, you too can experience the joy of physical media. There are even cassette tape cases for specific functions like play and pause or for specific playlists. Very nice.

magpi.cc/vinylemulator

Retro gaming NES frame

Video game art

Pixel art and old game cover art have their place in pop culture, as does modding the NES. Combine them and you have this framed NES cartridge cycling through art.

magpi.cc/nesframe

Learn robotics with Raspberry Pi

Build your own robot with Raspberry Pi, a few components, and these learning resources

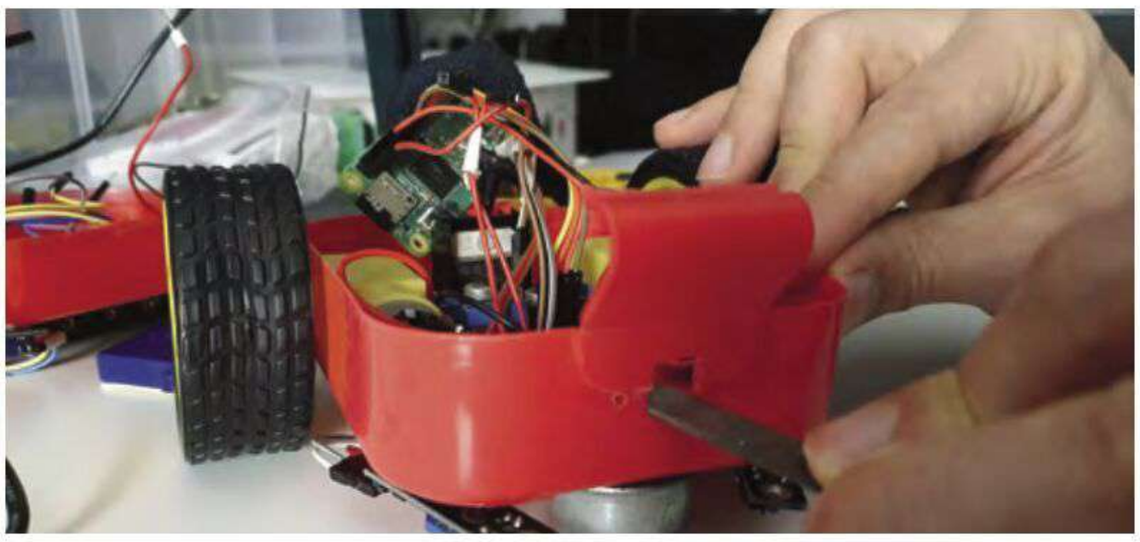
Build a low-cost Raspberry Pi robot

AUTHOR
Danny Staple

Price:
Free

magpi.cc/issues

If you are looking for a quick guide to building a small, low-cost wheeled robot, then we have you covered. Back in *The MagPi* issue #84, we started a series of tutorials on robotics.



Written by Danny Staple, the series kicks off with an overview of parts, such as the chassis, wheels, motors, a motor controller for Raspberry Pi, sensors, and power. It also looks at how to get the most performance at a low cost, with tips on choosing the right motors and gears. Subsequent tutorials cover putting the components together, controlling the robot through Python code, adding navigation abilities, and using sensors to detect and react to the environment. It works its way all up to adding basic artificial intelligence with OpenCV and a Camera Module.

Our tutorial series is a great way to get up-and-running with a robot, so pick up those back issues. The PDFs are available for free on our website.

Kits and courses

Keep things simple with a kit

SUNFOUNDER

We reviewed SunFounder's PiCar-V Kit V.20 back in *The MagPi* magazine #96, and were impressed with the huge amount of components in the box.

magpi.cc/picarv2

CAMJAM EDUKIT #3

This stalwart kit is the starting point for many Raspberry Pi robot makers. Wheels, motors, and wires are in the box (which you can use as a chassis).

magpi.cc/edukit3

MONSTERBORG

We love PiBorg's robots and the MonsterBorg is a sturdy, fast, and fun robot. It is a professional robot too, used for Formula Pi and RaceYourCode events.

magpi.cc/monsterborgreview

Build a robot buggy

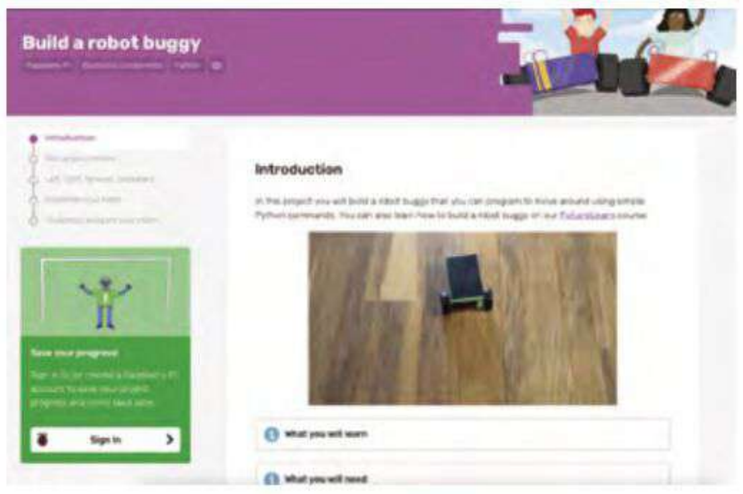
AUTHOR
Raspberry Pi Foundation

Price:
Free

magpi.cc/build buggy

We often hear people say they want to build a small robot, but whenever they start a course or tutorial it quickly becomes too complicated.

The 'Build a robot buggy' course by Raspberry Pi is designed to keep things as easy as possible. It introduces you to the parts you'll need (wheels, motors, chassis, etc.) and then guides you through setting it all up. It uses animated GIFs and lots of images to show what parts go where, and how to put them together (some soldering may be required). Then you use code to test out the parts, before putting everything together and programming your robot. The course is also available as part of FutureLearn (**magpi.cc/futurelearnrobots**) if you want to follow it as part of that.



Learn Robotics Programming, 2nd Edition

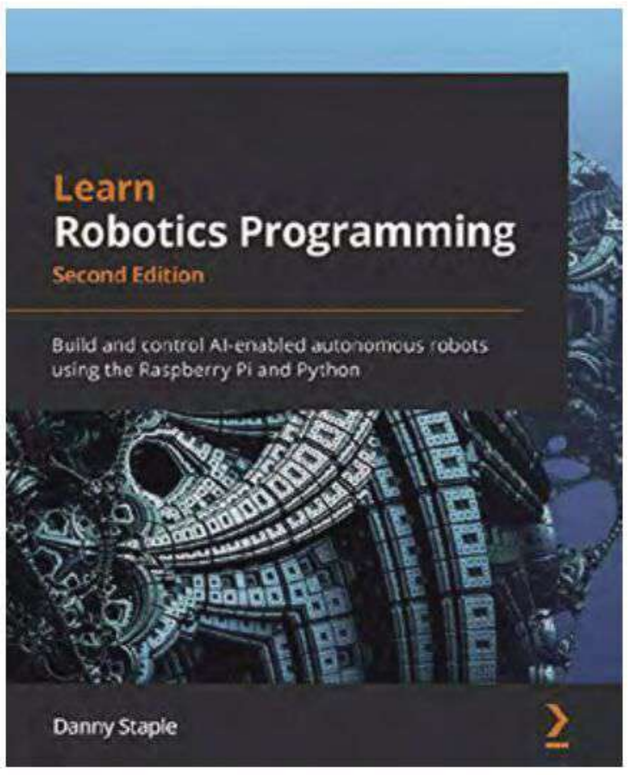
AUTHOR
Danny Staple

Price:
£30/$40

magpi.cc/robotprog2

If you have followed our tutorials on building a low-cost wheeled robot, then you'll be aware of the work of Danny Staple. If you want to take things further, then pick up a copy of the new book, *Learn Robotics Programming, 2nd Edition*.

This book goes through a similar process of building a low-cost wheeled robot in great detail. It looks into setting up a Raspberry Pi in headless mode to control the various wheels and interact with the sensors. As well as the basic controls, it has a deep dive into OpenCV and image interaction, along with voice communication, and also controlling a robot via a phone with Python.



Advanced robotics

Learn from industry and educational experts

MIT: INTRODUCTION TO ROBOTICS

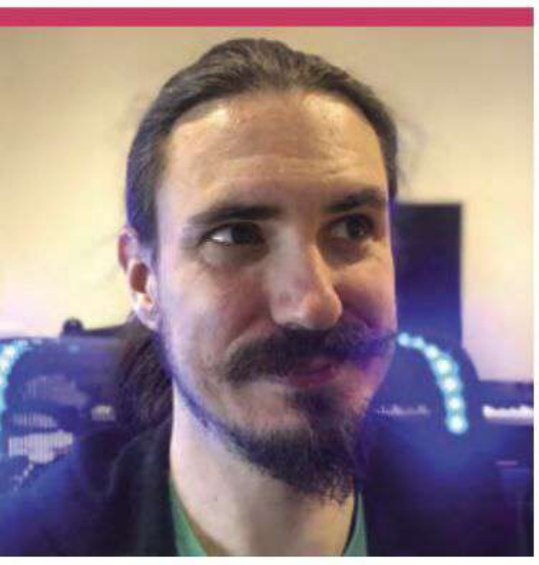
OpenCourseware rarely disappoints and this course provides an overview of robot mechanism, dynamics, and intelligent controls. It's a big step up from your first build, but the effort will be worthwhile.
magpi.cc/mitrobotics

STANFORD CS223A

Stanford's introduction to robotics is a detailed overview of the modelling, design, and planning of robot systems.
magpi.cc/standfordrobotics

COURSERA: MODERN ROBOTICS

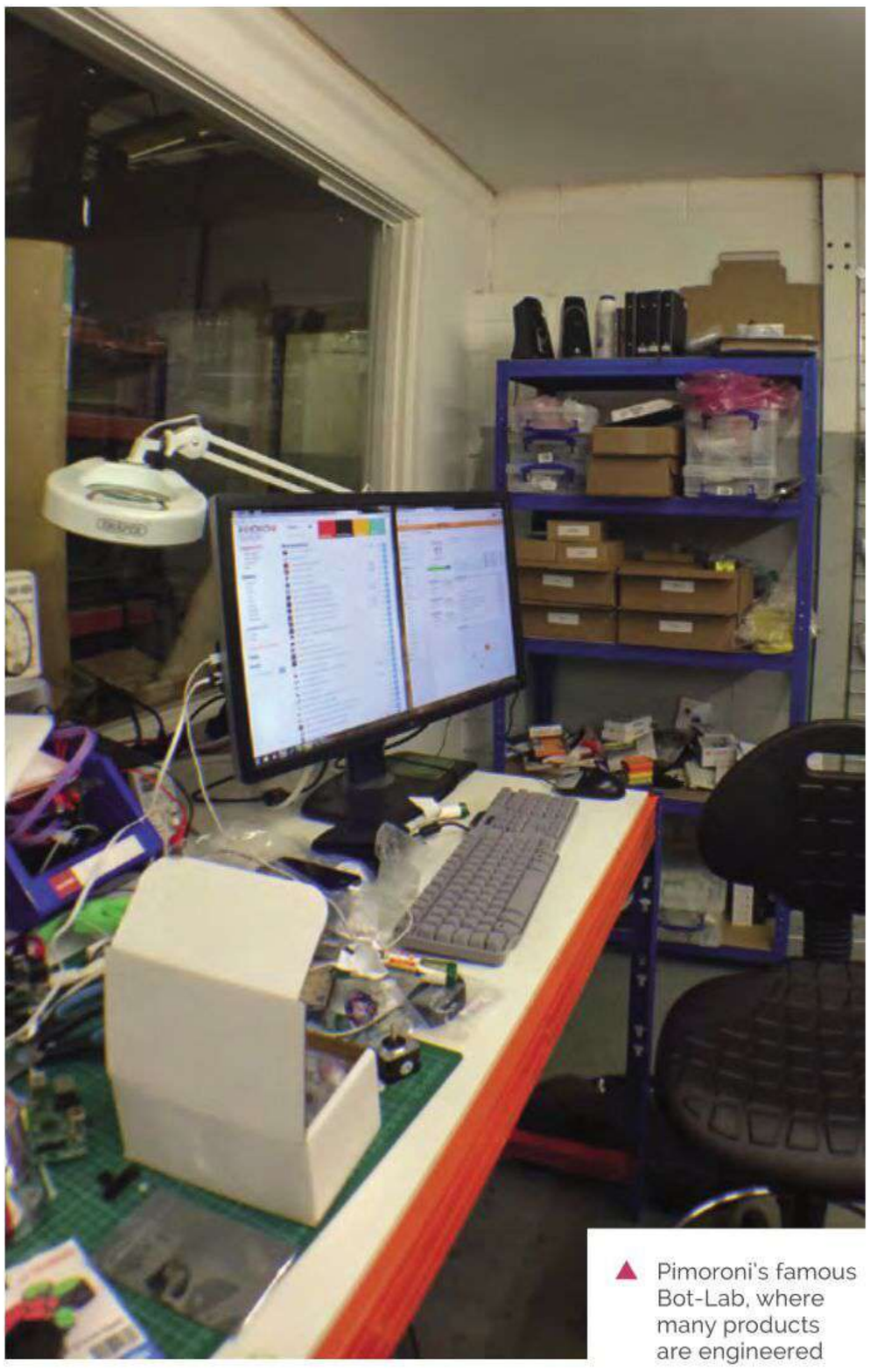
This course produced by Northwestern University covers kinematics, dynamics, and robot motion and control. Complete the course to get a certificate.
magpi.cc/courserarobotics



Phil Howard

From web designer to hardware engineer to software developer – professionally, Phil Howard's journey to Pimoroni is far from simple

- Name **Phil Howard** | Occupation **Software developer**
- Community role **Blogger** | URL **gadgetoid.com**



▲ Pimoroni's famous Bot-Lab, where many products are engineered

How did you start your Raspberry Pi journey? We've heard many stories, but not many that are quite like the case of Phil Howard, who you may know as Gadgetoid.

"I don't really have an electronics history to speak of," Phil tells us when we ask about his background. "Despite pulling things like broken Lego trains apart as a child and magically getting them to work, I'd never really picked up electronics tinkering as a hobby. About the closest I've come was taking an electronics A level for… about a week. I expected electronics tinkering and got a lot of maths. I didn't like maths. I took Media and Film studies instead. I've been spending the last 20 years slowly butting heads with the fundamentals I missed out on, but I think the film and media background helped put me where I am today. No regrets."

Phil had been programming since childhood, though, starting with Visual Basic and moving into classic web development languages like HTML and JavaScript. Taking an interest in the once-popular OpenPandora handheld Linux PC/game emulator, he was eventually introduced to Raspberry Pi and is now the lead developer over at Pimoroni.

When did you first learn about Raspberry Pi?

That's a tough one – it probably went around a news site, the OpenPandora forums, RSS feed, or word-of-mouth in the office. At the time I was working on web stuff for a marketing agency in Norwich, and had precious little background in electronics tinkering. I did have a keen interest in technology, however, and I'd been writing tech reviews casually. [I remember] the RS site allowed me to register interest, and my very first Raspberry Pi shipped on 19 November, 2012. I still have the shipping email.

Almost immediately after receiving [it], I started reaching out to manufacturers who were building add-ons, cases and, accessories – this was usually how I pulled together some blog content since getting 'accessory for X' was always

One of Phil's more recent projects was 32blit, a retro-style, programmable handheld games console

easier than getting 'X'. This was especially true with the immense demand for Raspberry Pi. These companies included AB Electronics and – yes – Pimoroni, and I think this might have been a small part of how I got their attention.

I quickly turned to blogging my adventures, writing reviews, and making myself heard in the Raspberry Pi community. You can still dig up my old ramblings at **pi.gadgetoid.com**.

What is it like working with/for Pimoroni?

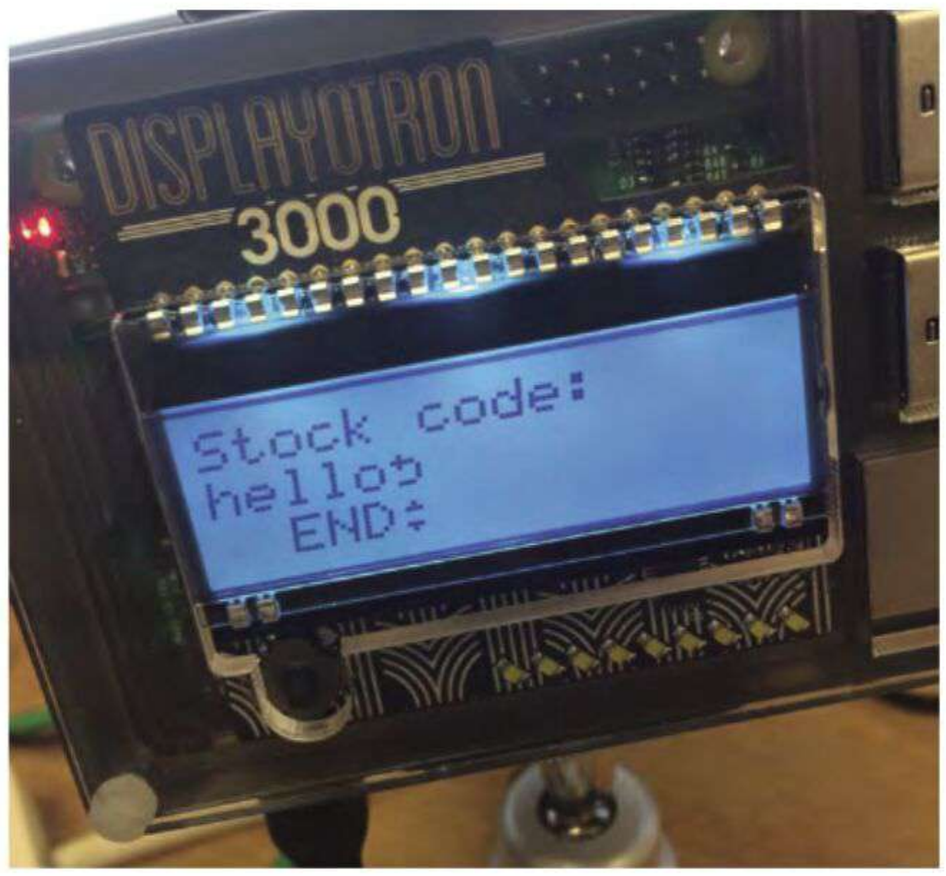
It's been quite the journey with the steady growth of Pimoroni, [and recently] I've run the full gamut from working at home, to uprooting my life to move to Sheffield and work with the Pirates, going through a building move or two, and finally uprooting again to relocate to Cambridge when working in offices became… not so popular.

When I first started, I was shooting the breeze with Jon – co-founder and CEO – over email and, probably, Skype at the time. We'd rubber-duck for each other through a new landscape of makey technical things, and we worked together on products like Display-O-Tron 3000 and Pibrella.

I'd travel back and forth to Sheffield and spend some days working more closely with them in their Burton Street warehouse, which comprised mostly of a narrow galley of laser cutters. It also had a haphazard stud-wall stock room, a packing/kitting/production area, an elbow-to-elbow engineering room we called (and still do) the 'Bot-Lab', and lots of roof leaks, freeze pops, and snacks… It felt a lot more like a shared hackspace than a workplace.

What are some of your favourite Pimoroni products you've worked on?

I think my favourite must be the ones I designed, prototyped, and programmed myself under the harsh, but fair, scrutiny of Jon and Paul. They gave me a lot of leeway to design a product, make mistakes, and wire things up wrong, but had a keen vision for how a Pimoroni product should pull together and were always nudging me toward that.



The Display-O-Tron 3000 was one of the many great Pimoroni projects Phil worked on

These include Pan-Tilt HAT, Micro Dot pHAT, Enviro pHAT (the very first one), and I also had a very deep involvement in the engineering behind Flotilla, touching every single PCB in Eagle (our house PCB editor of choice for better or worse) at one point or another. Flotilla was the first big project I worked on once I officially started working for Pimoroni, and it fused my past of HTML, CSS, and JavaScript with this new world of electronics. It was an extremely ambitious project and made for some haphazard, if very effective, on-the-job training. It was a stepping stone to my less hectic role writing Python software (and now Pico) for our wider product range.

This Month in Raspberry Pi

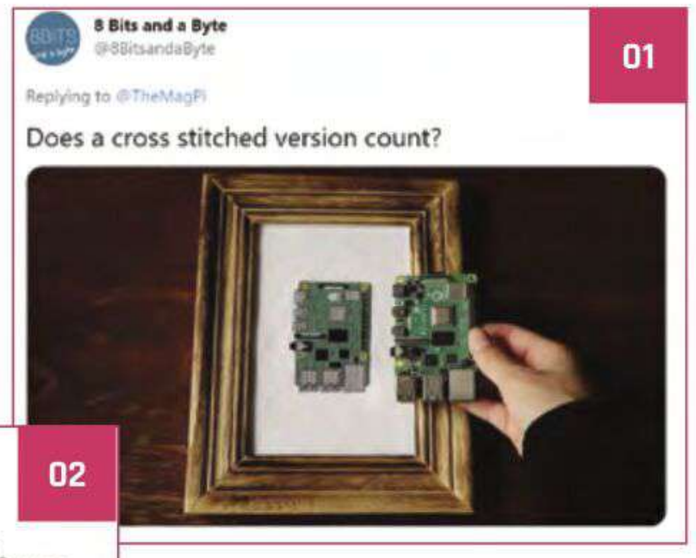
#MagPiMonday Showcase

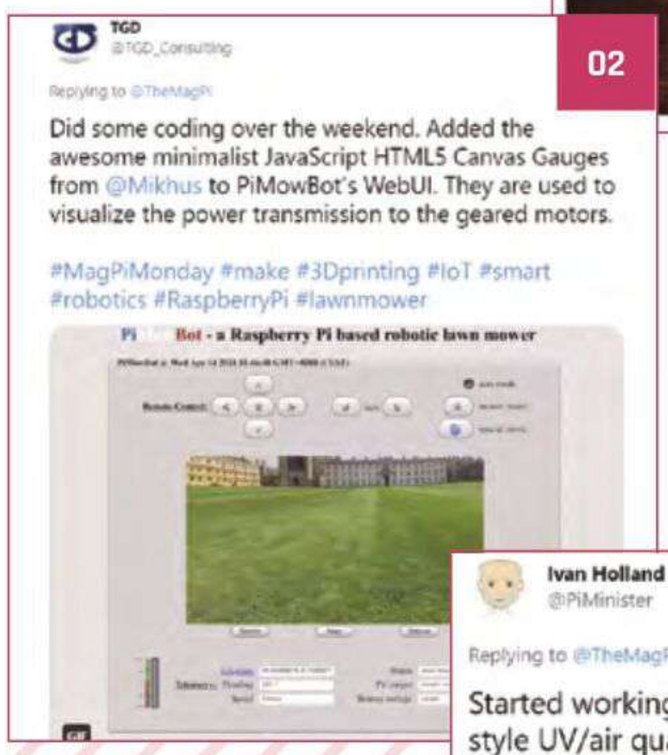
Amazing projects direct from our Twitter!

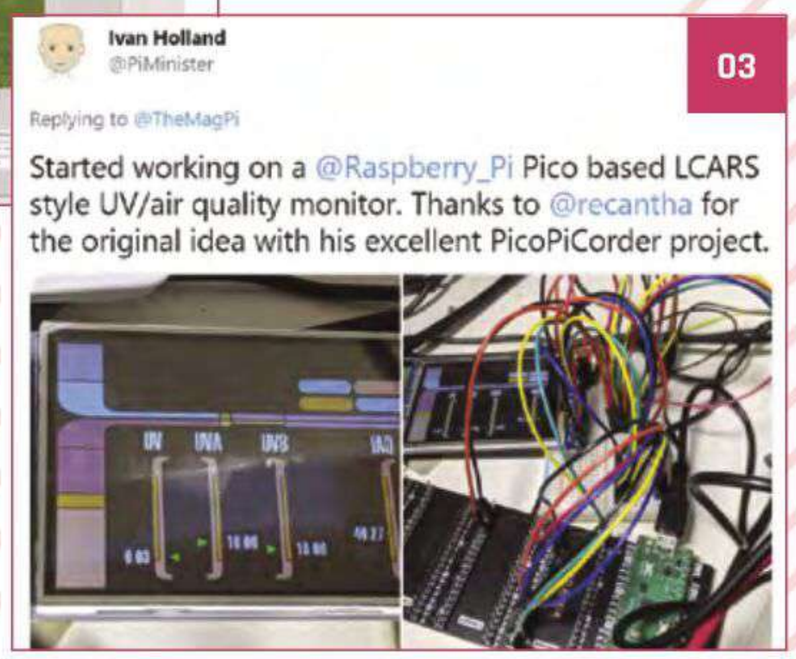
Every Monday we ask the question: have you made something with a Raspberry Pi over the weekend? Every Monday, our followers send us amazing photos and videos of the things they've made.

Here's a selection of some of the awesome things we got sent this month – and remember to follow along at the hashtag #MagPiMonday!

01. A cross-stitched version counts for us!
02. Your monthly PiMowBot update – this time, for its WebUI
03. We love a good LCARS interface. We're interested to see where this goes!
04. This 3D case is incredible, and the project inside it is very cool too
05. Calibrating thermometers can be tricky
06. We're seeing more and more of these project types lately!
07. This is a great visualisation of how GPIO works
08. PicoProducer looks very cool – we love the custom keycaps!
09. Kevin from diyelectromusic is always doing cool things with electronic sound
10. Can anything truly be tested for the British weather?
11. The finished PicoPicorder is the space sensor gadget of your DIY dreams







04

S Organ #E6E6FA
@makercupboard

Replying to @TheMagPi

My @Raspberry_Pi Pi-rate radio internet streaming radio. (Buttons still in progress). Very pleased with the 3d printed and painted case

06

Roberto
@roblamrod

Replying to @TheMagPi

Answering the doorbell from your mobile with a Raspberry Pi Zero W

Contesta el porterillo desde el móvil - IntercomPI
Mediante una Raspberry Pi Zero W y cuatro cosas más podrás contestar las llamadas del porterillo mediante tu móvil. Se ...
youtube.com

05

Replying to @TheMagPi

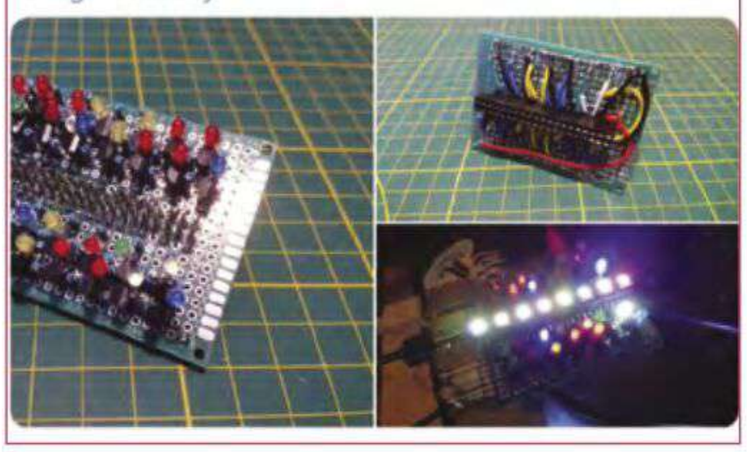
Home Assistant on a Zero W using BME680 break
Calibrating it here against other thermometers.
Intended to control heating, etc. in workshop.

07

Pierre-yves Baloche
@FunkyPiwy

Replying to @TheMagPi

As "seeing" what is happening on the GPIO can be of some interest, I've manually soldered about 300 points this weekend to build up this #RaspberryPi #PIOSpy visualizer. Tested out successfully with @pimoroni #blinkt! add-on. #MagPiMonday #Maker #Patience #MagPiTuesday



08

Pete Gallagher - Azure #MVPBuzz
@pete_codes

Slightly smaller keycaps for the #PicoProducer as the other ones were catching a bit...

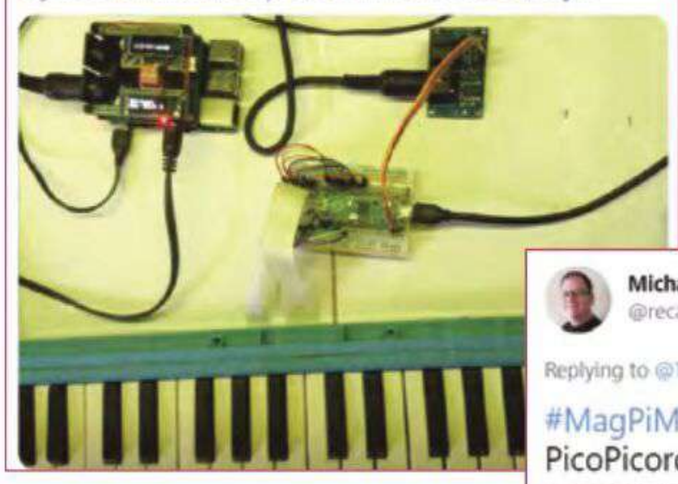
09

diyelectromusic
@diyelectromusic

Replying to @TheMagPi

I used a Pico to decode an old toy keyboard and use it to send MIDI to my #mt32pi.

diyelectromusic.wordpress.com/2021/05/02/toy...



10

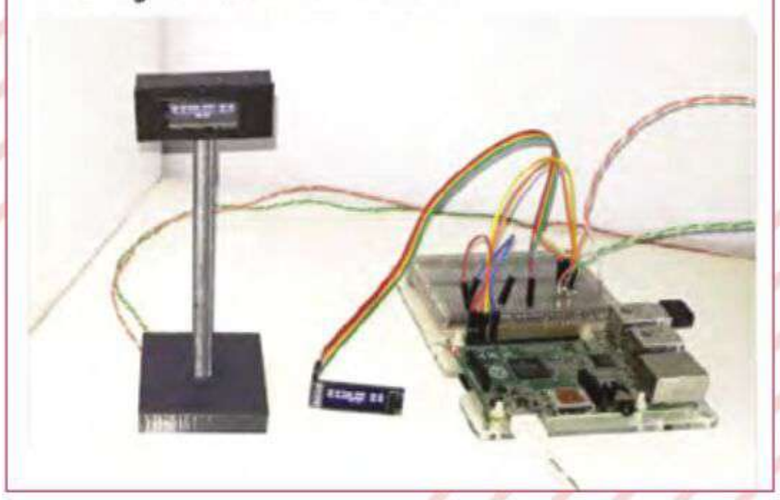
Stewart Watkiss
@stewartwatkiss

Replying to @TheMagPi

I created a model railway station departure board using a Raspberry Pi and a 0.91" OLED display.

penguintutor.com/projects/model...

Designed for a garden model railway, but not yet tested against the British weather.



11

Michael (Mike) Horne
@recantha

Replying to @TheMagPi

#MagPiMonday I've fully assembled and tested my PicoPicorder. Chock full of sensors and a thermal camera. Lots of blinkies! GPS still needs testing! Video attached.

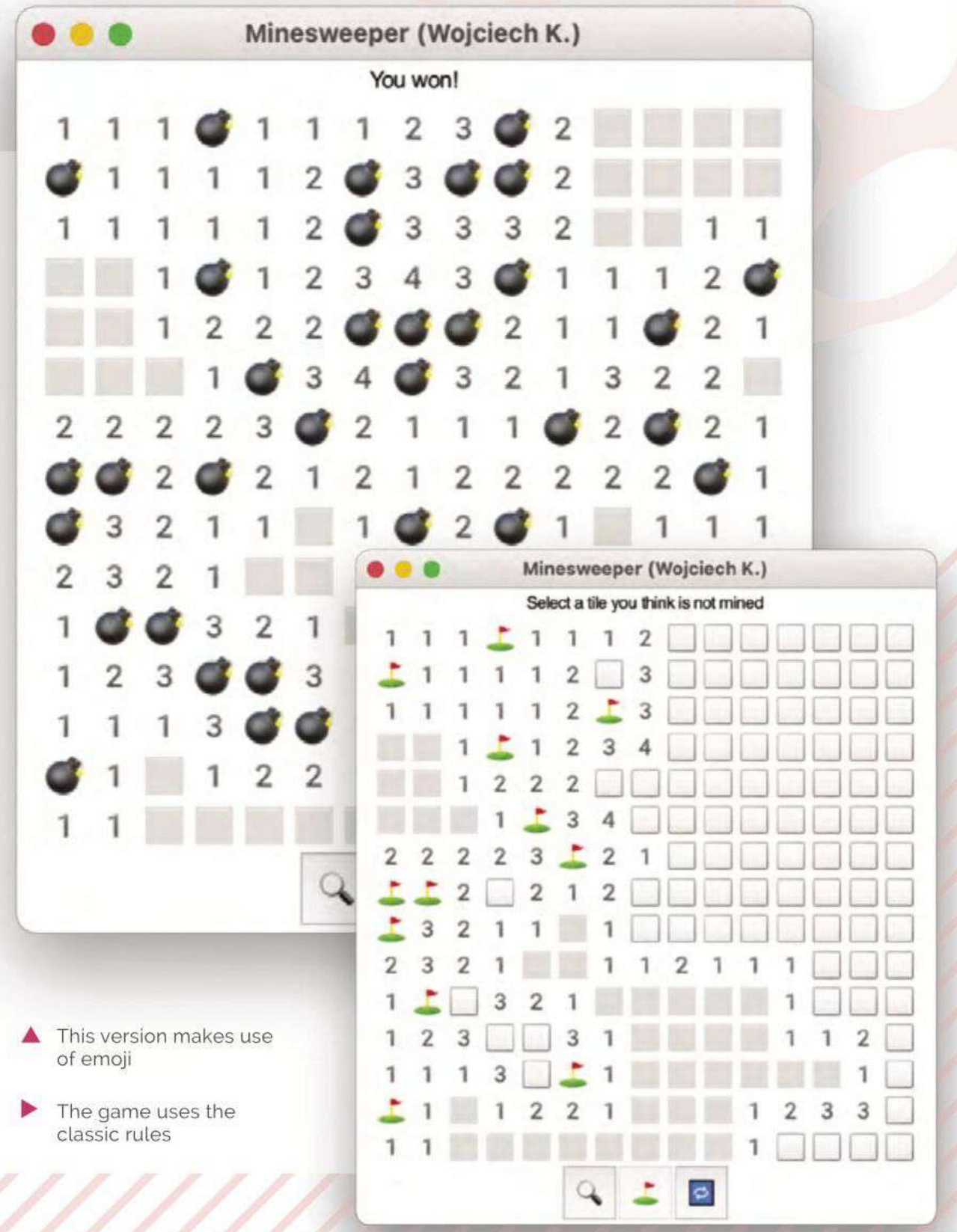
10:32 AM · May 10, 2021 · Twitter for Android

Design and success

Sometimes we receive emails from people with a cool project that we can't just fit into the project showcase section of the magazine. This month, we got an email from Mateusz Kosikowski about his 13-year-old son's successful attempt to create Minesweeper in Python.

"In the latest issue of the magazine is an article on creating GUIs in Python with the help of guizero," Mateusz tells us. "The topic got my son even more interested into programming. He even went through all the examples in the *Create Graphical User Interfaces with Python* [book]."

You can grab the code here from GitHub, which also has instructions on how you can run it: **magpi.cc/minesweeperpy**.



▲ This version makes use of emoji

▶ The game uses the classic rules

◀ It's a simple hardware build, but the code is a bit more complex, and was a learning curve for Chris

Pi-ve thermostat hack

Another email we got this month was about a cool little way of getting a Raspberry Pi Zero W to control a Hive Active Thermostat – from Chris, aka roadsnail. Bypassing the official cloud solution, it uses a Node-RED Dashboard for central heating and hot water control, including timers.

"The project touches upon several different technologies and open-source software: Zigbee wireless, MQTT message broker, Node-RED, and HTML," Chris says. "These all come together to produce Pi-ve. My main reason for making this was fun and education, and to put a Raspberry Pi Zero W that came with a previous subscription to *The MagPi* magazine to use.

You can find the code and other build details online here: **magpi.cc/pive**.

Crowdfund **this!**

Raspberry Pi projects you can crowdfund this month

AfterShock

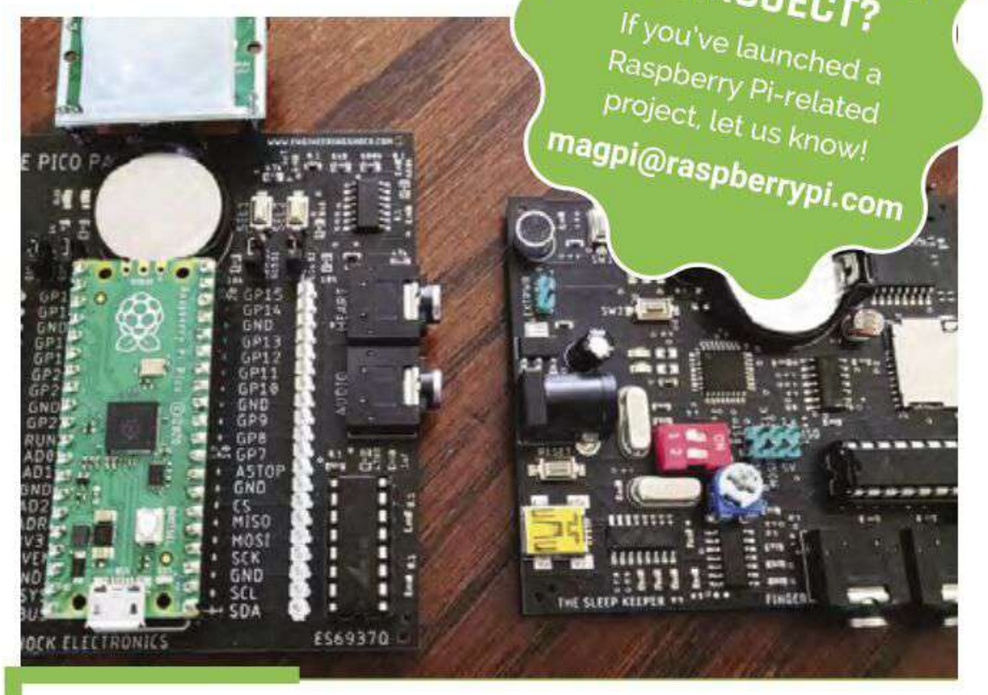
A different kind of earthquake sensor, this uses a seismic sensor instead of a seismograph, also has some smart filtering installed, and comes with some solar power options for Raspberry Pi as well.

▶ **kck.st/3uwel4S**

CROWDFUNDING A PROJECT?

If you've launched a Raspberry Pi-related project, let us know!

magpi@raspberrypi.com



Portable Pico Pal

A two-part Kickstarter for both Pico Pal and The Sleep Keeper, an Arduino-powered partner board. Pico Pal is a data logger that uses a Raspberry Pi Pico at its heart, and includes some battery backup as well.

▶ **kck.st/3f9309T**

Your Letters

First for Pico

So I have made my first Raspberry Pi Pico project. I have been using, and enjoying, Raspberry Pi since they came out. I still have some chuntering away doing what I want. I also wandered off into the ESP8266 and ESP32 world.

When [Raspberry Pi] Pico came out, I just bought a couple as you do… Well, this week I found a project for it: it was for my model railway and it was to automate a couple of points that were forever causing me accidents. When I changed the route, I would forget to reset them and the train would derail.

▼ Peter's project is camouflaged in these 3D-printed buildings

I had already found a setup that used reflected infrared to detect the trains. It used a PICAXE (another small microcontroller) to generate the pulses that a standard 38kHz sensor chip could reliably detect.

After I had built the prototype, it was apparent I would need more boards and it was a bit fiddly, so I decided "Why not use the Pico?" – I had two of them!

I started off with MicroPython and the 'Getting started with Raspberry Pi Pico' [guide]. I flashed LEDs, I said 'Hello World', I pressed buttons. Great. I then went on to program my 38kHz output and I could not make it fast enough. Oops! Was it because it is an interpreted language and bit slow?

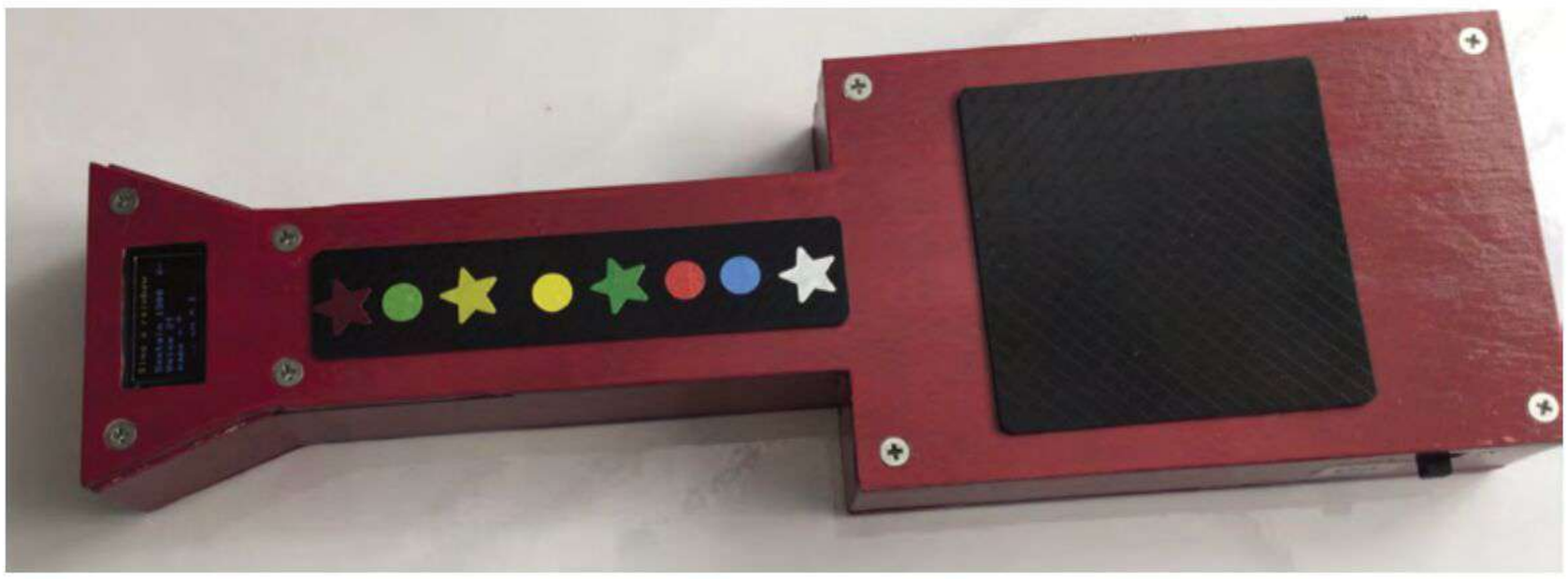
Never give in, Never surrender!

I then dug into the 'Getting Started with C/C++' and was able to generate my output – not only one stream but many more. So, job done, one Raspberry Pi [Pico] down, what's next?

Well, with the detector in its 3D-printed box, I can now 'see' the trains coming. All I have to do now is build and implement the logic to tell the points to change when the train is there.

Peter via email

We have been told that C programs run a bit faster on Pico compared to MicroPython, depending on the situation you need it for. Peter sent us some pictures of his build as well – at the time of writing, he's recently finished the build. We hope his trains stay on the rails.



▲ One of Mike's recent projects, this MIDI guitar is definitely a great one

Pi baker

I was excited when I first noticed that the legendary Mike Cook was writing for *The MagPi*. I'm not old enough to remember 'Body Building', but as a teenager tinkering in the mid-90s, his 'Run the RISC' articles were a source of ideas and inspiration and he responded to some of my questions via 'Rambles through Acorn Wood'.

I have less time for my own projects now (though my children watch videos in the car courtesy of a Raspberry Pi), but I work in the industry: I was a part of the project which produced the ARM Cortex-A53 processor used in Raspberry Pi 3.

So now I'm writing to say thank you to Mike, *The MagPi* and others (e.g. Maplin's 'Electronics' magazine) for their part in turning my interest in how things work into a career of making things work. They have contributed and continue to contribute to building an electronics/computing industry that makes life more exciting, interesting, and efficient. Long may it continue.

John via email

We're very lucky to have Mike build stuff for us every issue, he's inspired many people who have worked on *The MagPi* over the years. We're glad to know many readers have also been inspired by him.

Late for #MonthOfMaking?

I have been working on a project for a little while and I saw you had #MonthOfMaking a little while ago. Is it too late to send in my project? Will you do another month so I can submit it?

Zo via Twitter

It is always OK to send us your projects, whatever month it is. We love seeing what the community makes with Raspberry Pi and Raspberry Pi Pico, and we have many ways of sharing it with readers and the community at large. You can always show us on Twitter, Instagram, Facebook, or just email it to us.

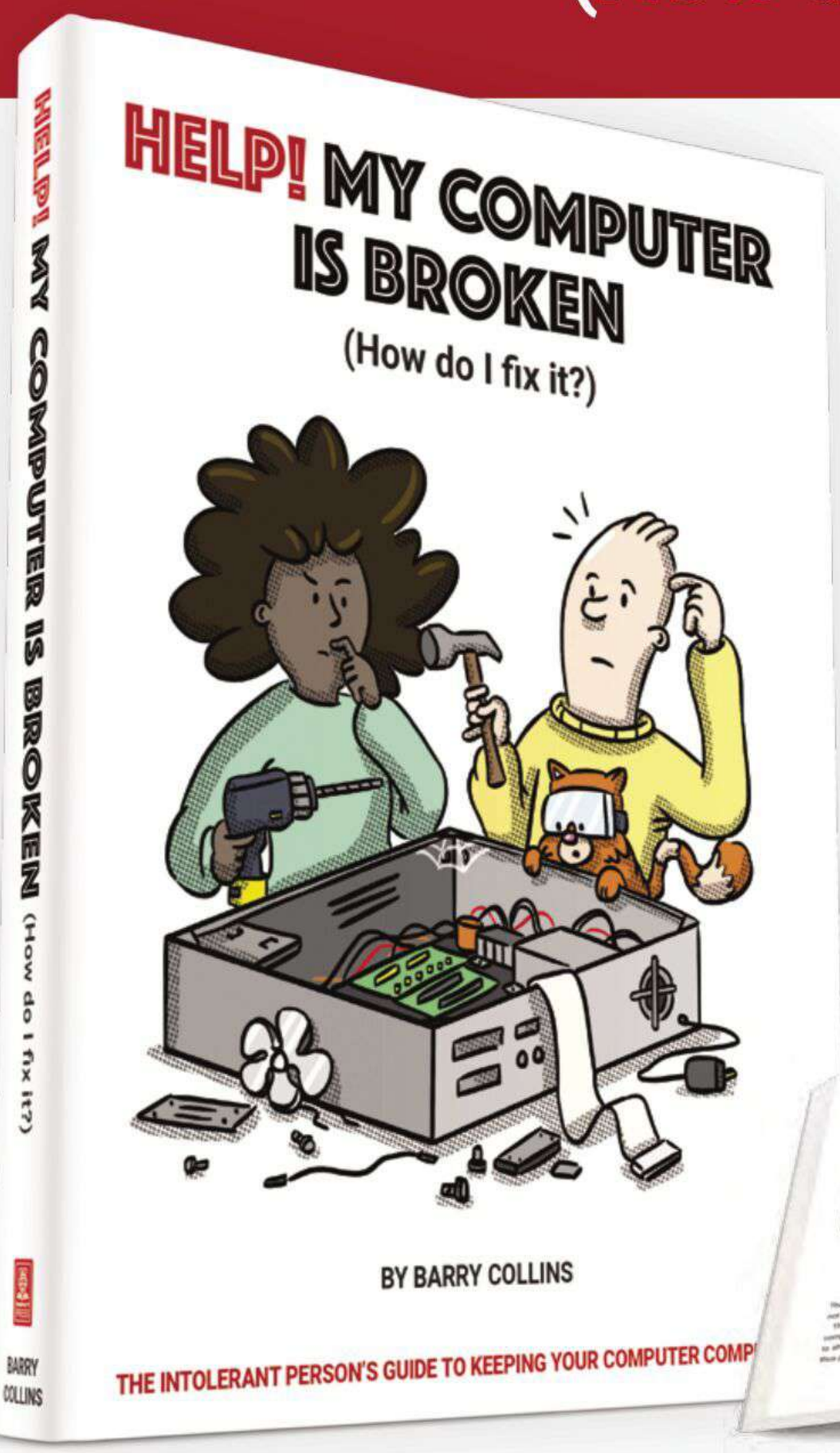
▲ #MonthOfMaking will return but in the meantime, don't stop showing us your projects

Contact us!

- Twitter **@TheMagPi**
- Facebook **magpi.cc/facebook**
- Email **magpi@raspberrypi.com**
- Online **raspberrypi.org/forums**

HELP! MY COMPUTER IS BROKEN

(How do I fix it?)

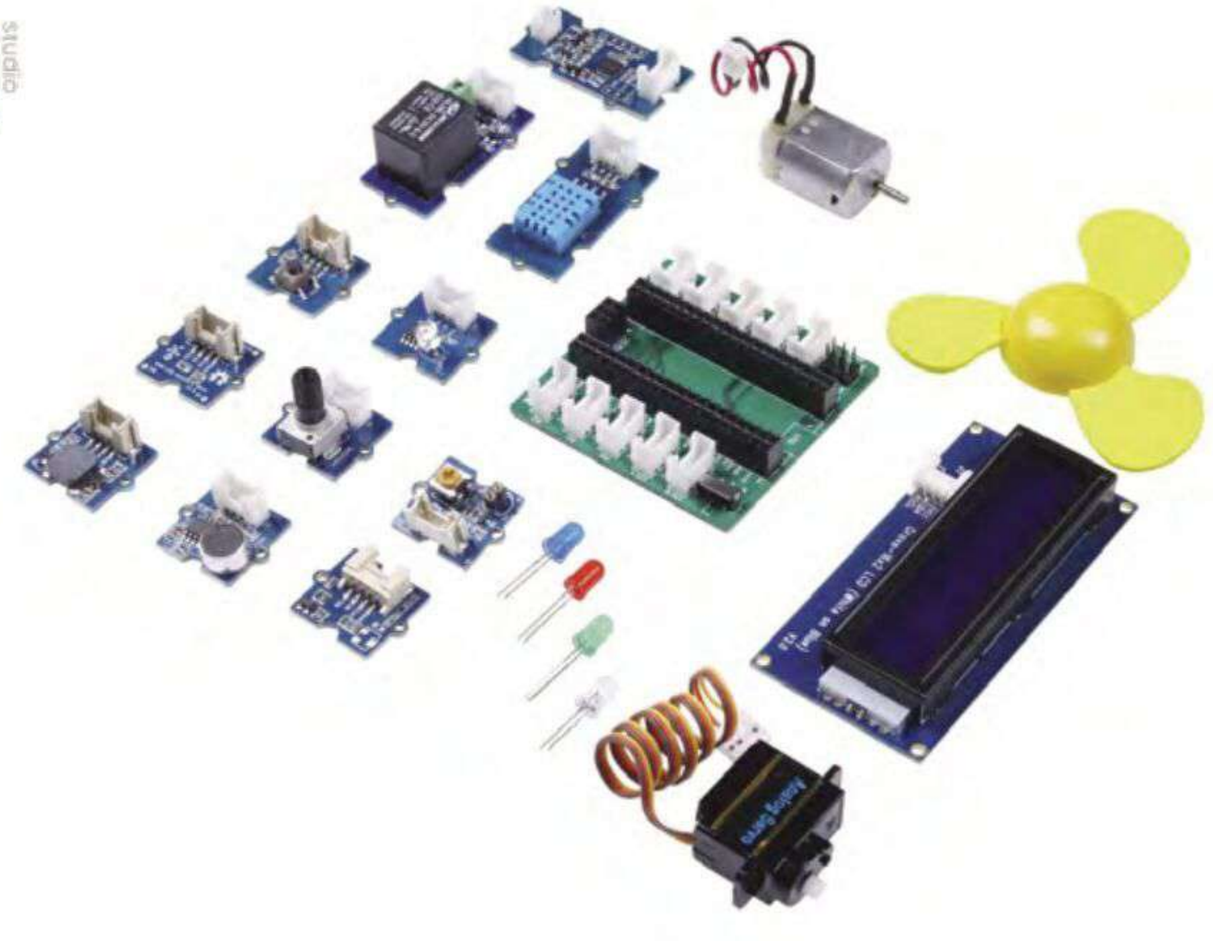


Help! My Computer Is Broken takes the most common computer problems and tells you how to fix them. It's as simple as that! If you've ever wondered why your laptop won't turn on, you can't get a WiFi connection, your printer isn't printing, or why everything is so slow – well, this is your book...

BUY ONLINE: magpi.cc/helpbook

WIN ONE OF FIVE GROVE STARTER KITS FOR RASPBERRY PI PICO

IN ASSOCIATION WITH seeed studio The IoT Hardware Enabler



This electronics starter kit is a great way to get to grips with how to program physical components using a Raspberry Pi Pico. We reviewed it in issue 105 and gave it 9/10 – now is your chance to win one.

Head here to enter: magpi.cc/win | Learn more: magpi.cc/grovestarter

Terms & Conditions

Competition opens on **26 May 2021** and closes on **24 June 2021**. Prize is offered to participants worldwide aged 13 or over, except employees of the Raspberry Pi Foundation, the prize supplier, their families, or friends. Winners will be notified by email no more than 30 days after the competition closes. By entering the competition, the winner consents to any publicity generated from the competition, in print and online. Participants agree to receive occasional newsletters from The MagPi magazine. We don't like spam: participants' details will remain strictly confidential and won't be shared with third parties. Prizes are non-negotiable and no cash alternative will be offered. Winners will be contacted by email to arrange delivery. Any winners who have not responded 60 days after the initial email is sent will have their prize revoked. This promotion is in no way sponsored, endorsed or administered by, or associated with, Instagram or Facebook.

HackSpace

TECHNOLOGY IN YOUR HANDS

THE MAGAZINE FOR THE MODERN MAKER

SUBSCRIBE AND SAVE UP TO 35% on the cover price

ISSUE #43

OUT NOW

hsmag.cc

Available on the App Store

GET IT ON Google Play

THE TOP RASPBERRY PI PROBLEMS FIXED

LOCATE AND FIX FAULTS WITH OUR GUIDE TO SOLVING COMMON ISSUES

THE MAGPI #107 ON SALE 24 JUNE

Plus!

Handheld builds

Make a weather watcher

Pico tablet

DON'T MISS OUT!
magpi.cc/subscribe

TWITTER	@TheMagPi
FACEBOOK	fb.com/MagPiMagazine
EMAIL	magpi@raspberrypi.com

EDITORIAL
Editor
Lucy Hattersley
lucy@raspberrypi.com

Features Editor
Rob Zwetsloot
rob@raspberrypi.com

Sub Editor
Nicola King

ADVERTISING
Charlotte Milligan
charlotte.milligan@raspberrypi.com
+44 (0)7725 368887

DESIGN
criticalmedia.co.uk

Head of Design
Lee Allen

Designers
Lucy Cowan, Sam Ribbits

Illustrator
Sam Alder

CONTRIBUTORS
Mike Cook, David Crookes, PJ Evans, Gareth Halfacree, Martin O'Hanlon, Rosemary Hattersley, Nicola King, KG Orphanides, Nik Rawlinson, Laura Sach, Mark Vanstone

PUBLISHING
Publishing Director
Russell Barnes
russell@raspberrypi.com

Director of Communications
Liz Upton

CEO
Eben Upton

DISTRIBUTION
Seymour Distribution Ltd
2 East Poultry Ave,
London EC1A 9PT
+44 (0)207 429 4000

SUBSCRIPTIONS
Unit 6 The Enterprise Centre
Kelvin Lane, Manor Royal,
Crawley, West Sussex, RH10 9PE
+44 (0)1293 312193
magpi.cc/subscribe
magpi@subscriptionhelpline.co.uk

recycle

CC BY NC SA

This magazine is printed on paper sourced from sustainable forests and the printer operates an environmental management system which has been assessed as conforming to ISO 14001.

The MagPi magazine is published by Raspberry Pi (Trading) Ltd., Maurice Wilkes Building, St. John's Innovation Park, Cowley Road, Cambridge, CB4 0DS. The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to goods, products, or services referred to or advertised in the magazine. Except where otherwise noted, content in this magazine is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0).

Raspberry Pi PRESS

ISSN: 2051-9982.

Build back **bolder**

Now is the time for makers to step up. By **Lucy Hattersley**

As I write, the world around us is springing back to life. I've been into town and had a wander around. It's good to see things starting to unfold after the long winter.

Computing often lends itself to quiet complementation, and the dark hours of winter are a good time to sit down and practise a skill: whether it's coding, crafting, or art. Prince used to say the long, bleak Minnesota winters were good for drumming practice. When you can't go outside, you focus on what's to hand.

Still. I feel a lot better for a little bit of sunshine and the chance to explore the great outdoors.

This newfound summer confidence showed up in our features this month. Obviously in the 'Summer Projects' feature, but perhaps even more pertinently in the 'Big Builds' one. If ever there was a time and place for 'thinking big', it's now.

The expansive nature of big builds – whether they're arcade machines, pinball tables, or photo booths – means they're sure to keep you busy over the summer months. But the clarion call is part of something much bigger. This year, have big ideas and make things happen.

We all need to build a better world in future years; one where we work in harmony with technology and make the world a better place for everybody.

Raspberry Pi was built to help children ('of all ages', as we like to say) learn how computers actually work. Unlike the glass, shiny tablets and phones; or sealed and glued laptops; Raspberry Pi has a layer of physicality where people get close to the metal: physically and mentally. You hold the board in your hand, see the components, and figure out how they work together to make the shiny stuff happen on the screen.

> This newfound summer confidence showed up in our features this month

And, of course, Raspberry Pi isn't just about looking at a screen. It's about interacting with the physical world through code. Sensors, motors, lights, and buttons are all used to get computers helping in the world.

Demystify it

When you demystify technology, it is no longer 'indistinguishable from magic'. Rather, it becomes a practical and useful tool that you can put to use to make the world around you a better place.

And that's what we hope people with Raspberry Pi will do. Just this month, the Farm Sensor Dashboard showed us how Raspberry Pi is helping farmers grow crops more efficiently. In recent months we've seen amazing projects like Real-time Bee Monitor, WeCount Traffic Sensors, and AI Noise Sensors. These projects are being used to monitor the activity of humans and nature together. These smart projects will help people make decisions to build better environments that work for us all.

These builds are fun (that bit is essential too), but they are also important. It is vital that we use technology to make the world a better place. Even in small, personal ways. So let's build back bigger, bolder, and use our technology with heart. Do no harm, try not to be evil, but do have a lot of fun.

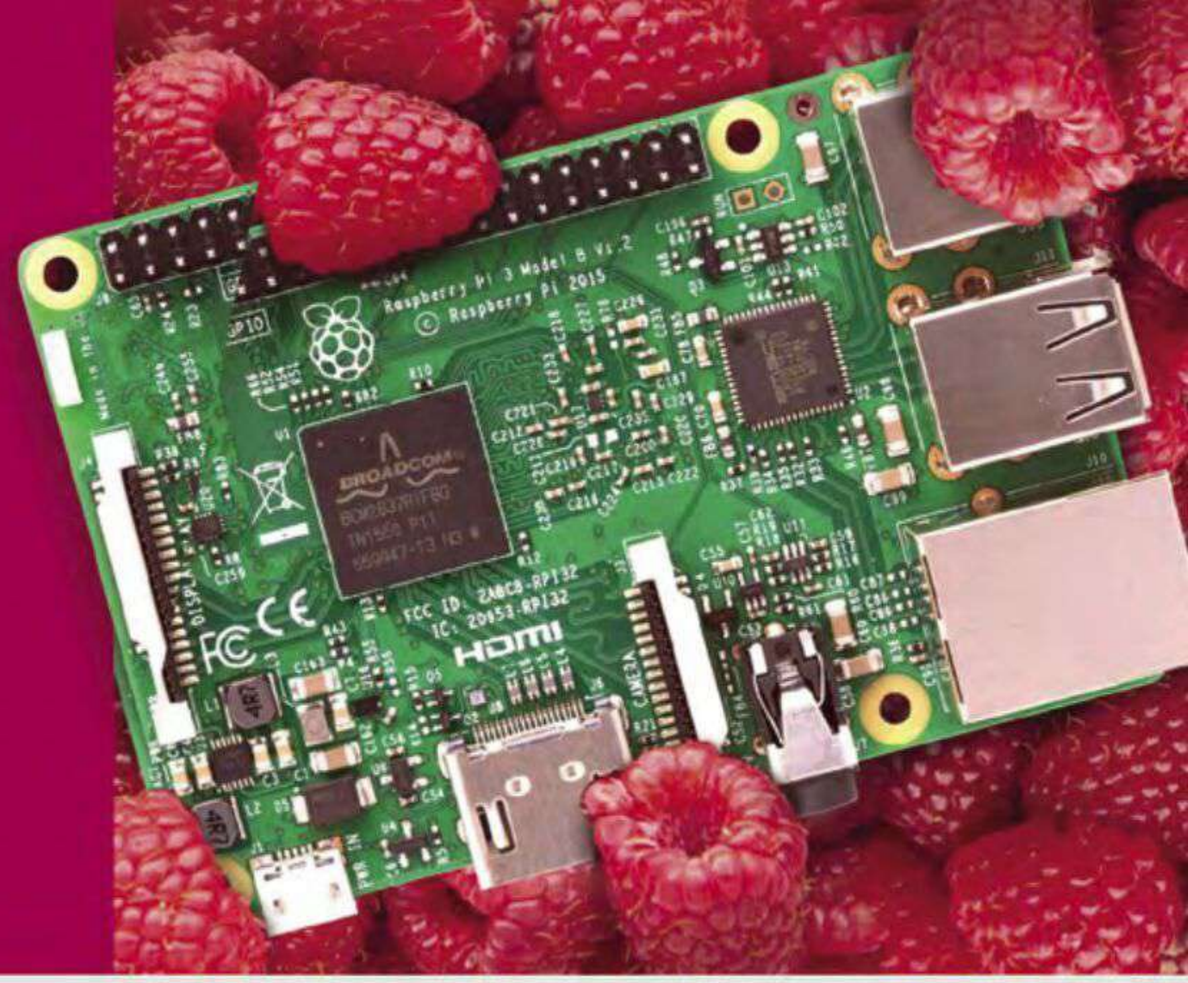
AUTHOR

Lucy Hattersley

Lucy is editor of *The MagPi* magazine and uses Raspberry Pi to monitor seismic activity in London and the health of her poor houseplants.

magpi.cc

American **Raspberry Pi** Shop



- Displays
- HATs
- Sensors
- Cases
- Arcade
- Swag
- Project Kits
- Cameras
- Power Options
- Add-on Boards
- Cables and Connectors
- GPIO and Prototyping

Partner and official reseller for top Pi brands:

adafruit · sparkfun ELECTRONICS · PIMORONI · HiFiBerry · and many others!

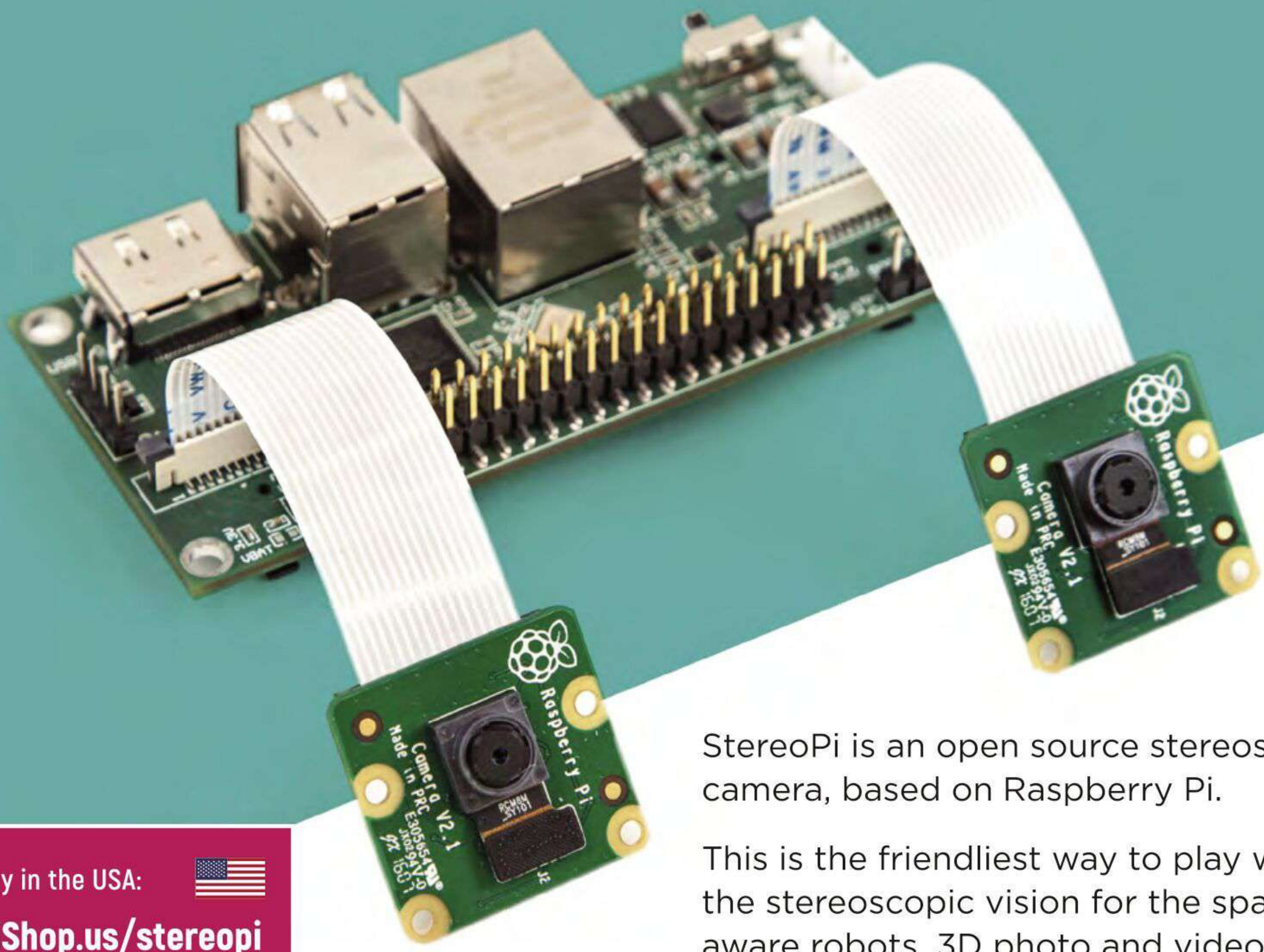
Price, service, design, and logistics support for **VOLUME PROJECTS**

USA PiShop.us

Canada BuyaPi.ca

Raspberry Pi APPROVED RESELLER

Do you know
HOW ROBOTS SEE?



StereoPi is an open source stereoscopic camera, based on Raspberry Pi.

This is the friendliest way to play with the stereoscopic vision for the spatially aware robots, 3D photo and video!

Buy in the USA:

PiShop.us/stereopi

Buy in Canada:

BuyaPi.ca/stereopi

RASPBERRY PI INSIDE

Raspbian

STOCK RASPBIAN SUPPORT

OPEN SOURCE

CROWDFUNDED PROJECT

LinuxGizmos.com
"The StereoPi can capture, save, livestream, and process real-time stereoscopic video and images for robotics, AR/VR, computer vision, drone instrumentation, and panoramic video."

MickMake
"With it you can do things like, stream stereoscopic 3D video to YouTube, build real-time depth maps using OpenCV, create panoramics using Hugin and even a 3rd person view of real life. Cool."

Raspberry Pi Blog
"There are some excellent community efforts too, of which our current favourite is this nifty dual camera board."

Hackster News
"You can hook this up to YouTube, to Oculus Go, you can use it with OpenCV.. I cannot wait to start messing around with these because it's basically a dream come true."

Elettronica In · LinuxGizmos.com · Geeky Gadgets · OPENELECTRONICS · electronics-lab